# Le periferiche del VIC vol. II

evm

# INDICE

# I N T R O D U Z I O N E

Questo secondo manuale sulle periferiche contiene tutte le notizie necessarie al funzionamento dei vari cartridges e le relative implementazioni disponibili sul VIC-20.

Essenzialmente e' diviso in 4 parti:

-HI-RESOLUTION GRAPHIC

-VIC-MON

-PROGRAMMER'S AID

-TAVOLE, PROGRAMMI, NOTE

Per quanto riguarda le prime 3 sezioni queste sono costituite dalle istruzioni d' uso dei principali cartridges disponibili sul VIC, mentre la IV sezione contiene tutte quelle tavole utili e quei riferimenti che possano rendersi necessari all' uso dei cartridges stessi.

Dei tre cartridges presentati i maggiori approfondimenti sono fatti sull' HI-RESOLUTION chiamato anche SUPER-EXPANDER oppure, in forma abbreviata HI-RES, e sul VIC-MON.
Sono riportate anche notizie e particolarita' sulle espansioni di emoria dato il verificarsi di problemi sia nell' utilizzo in combinazione fra loro dei vari Cartridges sia nell' uso di programmi spostati da una configurazione ad un' altra.
I semplici programmi riportati ci auguriamo che

possano essere di aiuto alla comprensione dei capitoli ed alle varie ed interessanti applicazioni che da questi possono derivarne.

Non esitate communque a contattarci per mezzo della scheda riportata al termine del presente manuale.

# INTRODUZIONE AL VIC-MON

## 1.1 Introduzione

Il VICMON e' il nome con il quale si identifica un monitor in linguaggio macchina costruito per consentire una facile messa a punto dei programmi in linguaggio macchina per il VIC 20 della Commodore.

Per non cadere in equivoci e' bene tenere presente che questa parte del manuale non vi insegnera' a programmare in linguaggio macchina o in Assembler, perche' per questo e' necessario fare riferimento ad altri tipi di manuali che vi segnaliamo:

MOS 6502 PROGRAMMING MANUAL

GUIDA AL PERSONAL VIC 20 -Ed. EVM

CORSO DI ASSEMBLER PER VIC 20 -Ed. EVM

Il terzo volume e' in corso di preparazione alla di data di stampa del presente volume

VICMON e quanto segue sono pertanto dedicati SOLO a coloro i quali hanno una certa familiarita' con la programmazione e conoscono i codici, le forme di indirizzamento e la logica della programmazione sia Basic che, almeno in parte del 6502, anche se dobbiamo sottolineare che non e' necessario essere degli esperti programmatori.
Tuttavia per semplificare il lavoro nell' uso di questo che resta un importante e delicato

strumento riportiamo in appendice le istruzioni relative al 6502 esemplificate anche nel manuale MOS nocnhe' una breve spiegazione.
Riportiamo anche, dal manuale GUIDA AL VIC 20 ediz. EVM, sia le tavole del Sistema Operativo che le tavole di salto alle routines Kernal.
Il programmatore che intende lavorare in Assembler e' bene che abbia una conoscenza approfondita delle funzioni relative almeno alle principali routines del S.O. sia per il risparmio di tempo e la ottimizzazione che ne ricava dai programmi stessi sia perche' almeno le routines di manipolazione del video sono indispensabili.

## 1.2 Sezioni VICMON

Questo manuale e' diviso nelle sottoindicate sezioni:

### PRIMA SEZIONE-INTRODUZIONE AL VICMON

Questa parte descrive il VICMON in termini generali.
Spiega come far partire il VICMON ed i termini convenzionali usati in questo scritto per la descrizione del formato dei comandi.

### SECONDA SEZIONE-I COMANDI DEL VICMON

In questa sezione e' spiegato dettagliatamente ogni comando di questa procedura, il suo formato, il suo uso e sono riportati alcuni esempi.
Questa sezione e' stata descritta in ordine

alfabetico relativamente ai comandi usati.

## 1.3 LE FUNZIONI DEL VICMON

Il VICMON consente le seguenti funzioni:

-Visualizzare una scelta area di memoria.

-Cambiare i contenuti di locazioni di memoria.

-Muovere blocchi di memoria.

-Riempire blocchi di memoria selezionati.

-Ricercare in memoria un determinato valore.

-Esaminare e cambia i registri principali.

-Fissare e rimuove i breakpoints

-Eseguire programmi in Linguaggio Macchina con controllo dei Breakpoints.

-Immagazzinare e ricerca sulle periferiche dati e programmi.

-Eseguire i programmi a diverse velocita' e con diverse modalita' selezionabili.

## 1.4 INIZIO E PARTENZA DEL VICMON

Il cartridge del VICMON deve essere inserito e rimosso dal VIC a macchina spenta.

Ricordiamo che deve essere inserito nella porta di espansione con la scritta rivolta in alto, con la massima cautela e senza cercare di forzare.

NOTA

Se si sta usando una MEMORY EXPANSION BOARD con alimentazioe separata ricordiamoci di spengere anche questa.

VICMON puo' essere usato in unione con un PROGRAMMER'S AID e/o con un HI-RESOLUTION GRAPHIC CARTRIDGE o altri aiuti.
Tuttavia, a parte la solita raccomandazione di spengere comunque la macchina e quanto altro in tensione, e' bene considerare che si possono avere dei conflitti operativi cambiando da un cartridge all' altro in particolare usando Board di espansione in cui i cartridge siano selezionabili con switch.

Nessun conflitto operativo invece usando VICMON con una qualsiasi espansione RAM sempre nella MEMORY EXPANSION BOARD.

Per iniziare ad usare il contenuto di questo cartridge, una volta eseguite correttamente le operazioni con le cautele dette e' sufficiente digitare:

SYS 24576 oppure SYS (4096*6)

premendo dopo il tasto di RETURN.

Ricordiamo che con questo comando si salta alla locazione indicata nel parametro dell' istruziòne SYS.

Lo schermo del VIC mostrera' ora i valori dei registri del 6502 a quella locazione di memoria nel seguente formato:

```
B      PC    SR  AC  XR  YR  SP
. ;   603E   33  00  63  00  F6
```

I registri visualizzati sono i seguenti:

PC = Program counter

SR = Stack register

AC = Accumulator

XR = X register

YR = Y register

SP = Stack pointer

Il Program counter riporta in esadecimale la locazione di memoria alla quale siamo saltati.

## 1.5 FORMATO DEI COMANDI

Molti comandi del VICMON sono di un singolo carattere alfabetico seguiti da un parametro se e' richiesto o se serve, e sono spiegati in dettaglio nella seconda sezione.
I parametri possono includere l' indirizzo di

partenza o l' indirizzo di partenza e di fine, il codice operativo o OP-CODE, gli operandi, i valori in esadecimale, ecc.

I comandi, tranne il comando J, sono eseguiti immediatamente dopo aver premuto il tasto di RETURN.
E' da segnalare che rimane in funzione l' editing tipico del VIC per correzioni ed aggiunte, per cui e' sufficiente riposizionarsi sopra i caratteri da correggere usando i cursori in modo diretto o con lo SHIFT, ma i comandi e le correzioni passano dal video al sistema operativo SOLO dopo il RETURN.

## 1.6 INDICAZIONI DI ERRORE

Qualsiasi errore nel quale siate incorsi durante la fase di INPUT sara' segnalato da un punto interrogativo (?) che segue la posizione dell' errore.
Come abbiamo detto si puo' correggere o riscrivere interamente facendo seguire da un colpo di RETURN.

NOTA

Lo scrolling dei listati, benche' possibile anche all' indietro (o SCROLLING UP)puo' dare dei risultati errati.

# S E Z I O N E   S E C O N D A

## I COMANDI DEL VICMON

### 2.1 Introduzione

In questa sezione ogni comando del VICMON viene presentato in ordine alfabetico e ne riportiamo un indice prima di addentrarci nell' esame dei singoli formati.
E' mostrato il fòrmato richiesto, lo scopo e la funzione.
Sono inclusi inoltre un piccolo esempio, la risposta che se ne ottiene dal sistema ed una spiegazione del risultato.

### 2.2 Simbologia e convenzioni

I parametri nei formati comando sono rappresentati secondo il seguente schema:

INDIRIZZO = due Bytes in forma esadecimale es. 0400.

DEVICE o PERIFERICA = un singolo Byte in esadecimale es. 08.

CODICE OPERATIVO o OP-CODE = un codice operativo in Assembler del 6502 es. LDA, JSR, ecc.

OPERANDO = un operando valido per la precedente istruzione del codice operativo es. $01.

VALORE = Un singolc Byte contenente un valore esadecimale es. FF.

DATA = Una stringa di dati letterali racchiusa fra parentesi o un valore esaedcimale. Successive voci devono essere separate da una virgola.

RIFERIMENTO = Un indirizzo di due Bytes es. 2000.

OFFSET o VALORE DI SALTO = Altro indirizzo di due Bytes.

I COMANDI : QUADRO RIASSUNTIVO

A = ASSEMBLE

B = BREAK POINT

D = DISASSEMBLE

E = ENABLE

F = FILL

G = GO

H = HUNT

J = JUMP

L = LOAD

M = MEMORY

N = NUMBER

Q = QUICK TRACE

R = REGISTER DISPLAY

RB= REMOVE BREAKPOINT

S = SAVE

T = TRANSFER

W = WALK

X = RETURN TO BASIC

## I COMANDI PER ESTESO

### A = ASSEMBLA

FORMATO : A (indirizzo)(op-code)(operando).

FUNZIONE : Assembla dei codici operativi partendo da un dato indirizzo.

Il comando consente di inserire, linea dopo linea, codici Assembler e di immagazzinarli in linguaggio macchina direttamente utilizzabile dal microprocessore.
L' indirizzo della successiva locazione di memoria disponibile oltre quello utilizzato dal codice operativo e dall' operando appena inseriti e' posto in attesa di un' altra istruzione.
Per far terminare la funzione A e' sufficiente premere il RETURN dopo l' inserimento dell' ultimo codice operativo.

Se viene inserito un codice operativo o un operando ILLEGALE il VICMON visualizzera' un punto interrogativo (?) prima della quantita' illegale e ritornera' alla funzione generale del monitor scrivendo un punto (.) in una nuova e successiva linea.

Se si dimentica di specificare un codice operativo o un operando, allora VICMON ignorera' la linea da assemblare e tornera' in ambito Monitor con un punto su una nuova linea.

NB. Ricordare che tutti gli operandi devono essere

dati in esadecimale preceduti dal segno dollaro ($).

ESEMPIO
Inserire il seguente gruppo di comandi:

```
LDA£$19
JSR$FFD2
RTS
```

con inizio all' indirizzo $1000

```
COMANDO: A 1000 LDA£$19 (RETURN)

SCHERMO: .A 1000 LDA£$19
         .A 1002

COMANDO: JSR$FFD2 (RETURN)

SCHERMO: .A 1000 LDA£$19
         .A 1002 JSR $ FFD2
         .A 1005

COMANDO: RTS (RETURN)

SCHERMO: .A 1000 LDA£$19
         .A 1002 JSR $ FFD2
         .A 1005 RTS
         .A 1006
```

RISULTATO
L' equivalente in linguaggio macchina del programma Assembler appena descritto e' stato immagazzinato in memoria dalla locazione $ 1000 alla $ 1005 inclusa.

N.B. Facciamo notare che l' Assembler del VICMON calcola automaticamente gli spazi necessari ad ogni codice operativo ed ai suoi operandi.

B = BREAKPOINT

FORMATO: B (indirizzo)

oppure

B (indirizzo),n

dove n e' un numero esadecimale di quattro digit che indica quante volte quell' indirizzo dovra' essere incontrato prima di un break.

FUNZIONE: fissa un punto di STOP in modo tale che un programma non sia eseguito per intero, ma si fermi ad una specifica locazione.

L' inserimento di un BREAKPOINT consente di far girare un programma fino ad un certo punto di uno dato indirizzo.
Se si usa in unione con un comando G (vedi dopo) il contenuto dei registri e' visualizzato automaticamente per controllare se la situazione degli stessi e' quella desiderata.
Se invece viene usato il modo Q per far girare il programma, si avra' una fermata al BREAKPOINT, ma i registri non verranno visualizzati.
In questo caso dovra' essere selezionato il modo W (anche per questo vedi dopo) e per visualizzare i registri sara' necessario premere il tasto RUN/STOP ed usare come vedremo il comando R.

E' da notare che il programma termina di girare PRIMA di eseguire l' istruzione all' indirizzo specificato nel BREAKPOINT, per cui e' necessario tenere conto che e' un comando NON INCLUSIVO. In altre parole non include l' ultimo indirizzo

Particolare cura deve essere posta nell' accertarsi che il BREAKPOINT non sia inserito tra il codice operativo ed i suoi operandi oppure in mezzo a dei data.
Non rispettando questa precauzione il comando B verra' ignorato con le conseguenze che cio' puo' portare.

E' possibile eseguire una istruzione un determinato numero di volte e quindi immettere un Breakpoint al passo seguente.
Si puo' fare cio' specificando il numero delle iterazioni cioe' a dire il numero di cicli da eseguire tramite il parametro N come abbiamo detto all' inizio in relazione al formato dell' istruzione.
Se non e' dato nessun numero dopo l' indirizzo il programma si fermera' quando incontra quel dato indirizzo per la prima volta.
Per immettere un BREAKPOINT si tratta semplicemete di scrivere B e di premere il RETURN, si eseguira' poi il programma tramite un comando G o Q.

ATTENZIONE

E' bene ricordare che se il programma prima di arrivare al BREAKPOINT incontra un salto che lo mandi ad eseguire un altro programma o una subroutine, fin quando non si torna indietro il comando B non verra' eseguito.
Ed e' bene valutare anche il caso che se il salto e' condizionato PUO' darsi ANCHE che non si

incontri mai il BREAKPOINT.

Si puo' usare un solo comando B prima che il programma inizi a girare, mentre quando questa funzione e' stata eseguita, se si desidera, prima di far ripartire il programma si puo' inserire un' altro, comando di BREAKPOINT, ma sempre uno solo per volta.

Se per quanto scritto nella nota precedente o per qualsiasi altro motivo non si giunga a nessun BREAKPOINT, l' esecuzione del programma puo' essere fermata tramite il tasto di RUN/STOP insieme al tasto di RESTORE.
Questa operazione ci portera' pero' di nuovo in ambito BASIC.
Sara' necessario reinizializzare il VICMON con il SYS 24576 e cercare di isolare il problema fissando un nuovo BREAKPOINT.

I ESEMPIO
Ipotizziamo di avere in memoria un programma che parta dalla locazione $100 e vada fino alla locazione $1200 e si desideri arrestarne l' esecuzione prima della linea di locazione $1050.
Dovremo allora scrivere:

COMANDO: B 1050 (RETURN)

N.B.
Ricordiamo ancora una volta che se avessimo voluto fermarci alla $1050 COMPRESA avremmo dovuto scrivere B 1051.

Si prosegue con:

COMANDO: Q 1000 (RETURN)

RISULTATO
Il programma sara' eseguito a velocita' lenta, fermandosi prima della linea 1050 e resteremo in modo WALK ( vedi dopo per gli effetti e l' uso di WALK).

II ESEMPIO
Si desidera fissare un BREAKPOINT in modo tale che il programma si fermi dopo aver effettuato l' istruzione di cui alla linea $1100 per la terza volta.

COMANDO: B 1100, 0003 (RETURN)

COMANDO: G 1000 (RETURN)

RISULTATO
Ricordando che l' uso dei comandi G o Q e' una libera scelta dell' utente, il programma partira' dalla linea 1000 e si fermera' dopo aver eseguito i comandi di cui alla linea 1100 le tre volte richieste.
Successivamente verra' visualizzato il contenuto dei registri del 6502 nel formato indicato all' inzio di questa parte, ma con il PC che sara' alla linea 1000.

D = DISASSEMBLA

FORMATO: D(indirizzo)

oppure
D(indirizzo di partenza),(indirizzo di fine)

FUNZIONE: Questo comando serve per disassemblare programmi, routines o in generale gruppi di codici a partire da un certo punto della memoria oppure fra due indirizzi specificati nella seconda parte del comando.

Il comando D consente di riconvertire i codici presenti nella memoria del computer e quindi in formato binario ( anche se ricordiamo che vengono visualizzati byte per byte in forma esadecimale), nel corrispondente linguaggio ASSEMBLER.
Si puo' specificare un linguaggio di inizio, nel qual caso verra' disassemblata la linea corrispondente a quell' indirizzo.
In questo modo il sistema restera' in ambito del comando DISASSEMBLER e si potra' usare il cursore per disassemblare le altre linee.
Usando infatti la funzione CURSOR-DOWN saranno disassemblate le linee successive alla prima, mentre con il CURSOR-UP quelle precedenti.
E' tuttavia da notare un particolarre e cioe' che queste funzioni NON inizieranno fin quando il cursore non si trovera' o in cima o in fondo allo schermo.
Questa funzione e' tipica del Sistema Operativo del VIC ed infatti risultati simili, pur ovviamente con altri comandi, si ottengono anche in ambito BASIC.

ATTENZIONE

Facendo eseguire questi scrolling in alto o in basso con il cursore si possono NON ottenere dei risultati validi a causa dell' inaccurata traduzione dei codici dal linguaggio macchina all' ASSEMBLER. Cio' lo abbiamo notato in particolare usando la funzione SCROLL-UP.

In alternativa si puo' specificare la parte di memoria da disassamblare. In questo caso le linee specificate saranno visualizzate sullo schermo una dopo l' altra.
Naturalmente se le linee da disassemblare sono troppe rispetto alla capacita' dello schermo, il relativo contenuto scorrera' verso l'alto.
Per fermare lo scrolling e' necessario premere il tasto di RUN/STOP.
Con questa operazione si resta in ambito DISASSEMBLER, infatti questa funzione puo' essere continuata con il tasto CURSOR-DOWN.

Quando ci troviamo in ambito Disassembler una linea di codice puo' essere modificata o riscritta usando l' editor del VIC 20, cioe' semplicemente riposizionandoci sopra e riscrivendola da capo. Ricordarsi poi di premere il RETURN.
Usando questo sistema si attiva automaticamente il comando A per l' assemblaggio.
Qualora si sia entrati in modo Assembler il cursore rimane posizionato dopo l' indirizzo sulla linea seguente la linea corretta.
Per uscire dal modo Assembler eseguire un clear di schermo (ricordiamo che si fa premendo contemporaneamente il tasto di SHIFT e quello di CLR/HOME) e dopo premere il RETURN.

ESEMPIO

Si desideri disassemblare le linee di codice macchina inserite nell' esempio sull' utilizzo del comando ASSEMBLER visto in precedenza e cambiare l' indirizzo della seconda linea da FFD2 a FFDO.

COMANDO: D 1000,1005 ( RETURN)

```
SCHERMO: . 1000 LDA £$19
1002 JSR $ FFD2
         . 1005 RTS
```

AZIONE: posizionare il cursore in modo da essere sul 2 della scritta FFD2.

SCRIVERE: O (RETURN)

```
SCHERMO: . 1000 LDA £$19
         .A1002 JSR $FFDO
         .A1005 RTS
```

RISULTATO
Il codice macchina e' disassemblato dalla locazione $1000 alla $1005.E' stato eseguito il cambiamento richiesto e successivamente immesso in memoria con il tasto RETURN.
Come detto in precedenza si puo' a questo punto uscire dal modo ASSEMBLER.

E = ENABLE cioe' abilita una pagina zero virtuale

FORMATO: E(indirizzo)

FUNZIONE: Fissa a parte una pagina zero virtuale in modo tale che il VICMON non interferisca con le eventuali variabili dell' utente

VICMON usa le locazioni da $00 a $71 della pagina zero della memoria.
Le Kernal Routines del Sistema Operativo usano le restanti locazioni della pagina zero ed altre ancora che vanno dall' indirizzo $0200 a $0800.
Poiche' il vostro programma puo' assegnare variabili che saranno immagazzinate in pagina zero, nella sua esecuzione, lo stesso programma puo' interferire con alcune informazioni ivi inserite in precedenza.
Per prevenire la distruzione delle informazioni contenute in questa importante zona di memoria, il comando E consente di fissare una pagina zero VIRTUALE di 256 Bytes in altra parte della memoria RAM disponibile.
Quando e' stata fissata una pagina ZERO virtuale ed inizia a girare il programma, allora VICMON scambia automaticamente i contenuti della pagina ZERO NORMALE con i contenuti della pagina ZERO VIRTUALE, proteggendo cosi' sia lo stesso VICMON che come abbiamo detto adopera routines in questa pagina e gli indirizzi delle altre routines del Sistema Operativo.
Al termine dell' esecuzione del programma i contenuti della pagina ZERO vengono ripristinati.
Quando poi il programma girera' correttamente e non sara' quindi piu' necessario adoperare questo trasferimento sara' sufficiente disabilitare il comando semplicemente con:

E 0000

facendo ricoincidere la pagina ZERO normale con quella virtuale.

ESEMPIO
Si desideri fissare una pagina ZERO virtuale con indirizzo di inizio $1000.

COMANDO: E 1000(RETURN)

RISULTATO
Le locazioni da $1000 a $10ff sono definite come zona della pagina zero virtuale

F = FILL memory(riempi la memoria)

FORMATO: F(indirizzo di partenza),(indirizzo di fine),(valore)

FUNZIONE: Riempe la memoria contenuta fra due specificati indirizzi con un dato valore.

Il comando F consente di inserire un valore NOTO entro uno specifico blocco di memoria.

Cio' puo' essere utile per inizializzare una struttura di dati o per ripulire il contenuto di un' area di memoria.
Per questo motivo il comando F deve contenere tutti i parametri enunciati e cioe' l' indirizzo di partenza, l' indirizzo di fine ed il dato da caricare nella memoria compresa fra questi due indirizzi.

Il dato deve essere sempre espresso in forma esadecimale.
Naturalmente, dato che trattasi di un lavoro a blocchi non si devono usare le locazioni da $0000 a $01FF cioe' la pagina ZERO e UNO della memoria del VIC 20 senza usare particolari protezioni come ad esempio quella vista in precedenza.

ESEMPIO
Si desideri scrivere il dato $EA (cioe' una istruzione cosi' detta NON OPERATIVA ) dalla locazione $1000 alla locazione $2000 inclusa.

COMANDO: F 1000,2000,EA (RETURN)

RISULTATO
L'istruzione non operativa EA e' stata immediatamente scritta nelle locazioni richieste.

G = GO (vai)

FORMATO: G

oppure

G(indirizzo)

FUNZIONE:Serve per far incominciare a girare un programma partendo dalla attuale. locazione contenuta nel Program Counter, oppure, nel secondo formato, iniziando da uno determinato indirizzo.

Il comando G puo' essere usato da solo o unitamente ad un indirizzo di partenza.
Nel primo caso il VIC eseguira' il programma in memoria o la subroutine del Sistema operativo iniziando dalla locazione contenuta in quel momento nel Program Counter.
E' necessario fare attenzione nell' uso delle subroutines del Sistema Operativo perche' non conoscendone bene il loro uso e' facile entrare in un LOOP o ciclo infinito.

NOTA
Le subroutines del S.O. del VIC 20 sono riportate per esteso in un manuale edito dalla EVM " GUIDA AL VIC 20".

Per visualizzare il contenuto dei vari registri ed in questo caso ricordiamo che puo' essere importante vedere l' indirizzo contenuto nel Program Counter usare il comando R come spiegato in seguito.
Nel caso invece si usi G seguito da un indirizzo, allora l' esecuzione del programma partira' dalla locazione di memoria specificata dall' indirizzo dell' istruzione stessa.
Il comando G riporta i registri al loro ultimo valore conosciuto e nel caso sia attivato l' uso della Pagina ZERO virtuale come descritto in precedenza con il comando E, allora verra' cambiata immediatamente la pagina ZERO virtuale con quella normale.

L' esecuzione dei programmi continuera' fino al BREAKPOINT come abbiamo visto prima, sempre che ne esista uno o fino alla fine del programma con le condizioni ed i risultati che esamineremo.
Se l' esecuzione del programma viene arrestata da un BREAKPOINT allora verra' anche visualizzato il contenuto dei registri con i valori presenti in

quel momento, cioe' dopo l'elaborazione.
Se il programma termina con un RTS (ReTurn from Subroutine, cioe' ritorno da subroutine) allora torneremo in ambito Basic.
Se invece l' ultimo comando incontrato nel programma ( che quindi non e' necessariamente la fine del programma stesso) e' un BRK(BreaK), allora resteremo in ambito VICMON.
Nel caso che non esista nessuna istruzione di termine programma o di STOP o comunque non si verifichi nessuna condizione di fine sara' necessario interrompere l' esecuzione altrimenti infinita, con il tasto di RUN/STOP e RESTORE.
Come detto prima usciremo dall' ambito VICMON per tornare in BASIC, ma potremo anche rientrare in VICMON senza perdere il programma.

NOTA.

Se nel vostro programma ci sono state variazioni al colore di schermo o sia stato cambiato il colore delle lettere puo' verificarsi il caso che non riusciate a leggere i registri visualizzati o la scritta READY.
Ritornandoci sopra con il cursore riusciremo pero' a visualizzarne il contenuto.
Inoltre l' uso frequente di BREAKPOINT, in particolare alle prime prove di esecuzione dei programmi previene la possibilta' di incorrere in dei LOOP o almeno ne diminuisce il rischio.
Come abbiamo detto questo non e' l' unico metodo per far girare i programmi in ambito VICMON.Per gli altri metodi vedi le sezioni relative ai comandi J,W,Q.

ESEMPIO
Ipotizziamo di avere un programma in memoria e di

desiderare che esso vada in esecuzione a partire non dall' inizio ma dalla linea presente alla locazione $2000.

COMANDO: G 2000(RETURN)

RISULTATO
I registri vengono ripristinati. Il Program Counter viene fissato a $2000. Se e' stata scelta una pagina ZERO virtuale e' in questo momento che VICMON effettua lo scambio.
Dopo aver fatto questo il programma inizia la sua esecuzione a partire dalla istruzione contenuta in $2000.

H = HUNT (ricerca)

FORMATO: H(indirizzo di partenza), (indirizzo di fine),(dati).

FUNZIONE: Cerca in un blocco di memoria specificato dagli indirizzi dei dati o delle stringhe di caratteri.

Il comando HUNT localizza ogni selezionato gruppo di caratteri in memoria e li visualizza sullo schermo.
Si puo' utilizzare questo comando per localizzare dati che devono essere espressi in forma esadecimale o per trovare stringhe di caratteri di una lunghezza massima di 88 caratteri.
Le stringhe devono essere specificate in forma

letterale e precedute dal segno '(accento).
Le locazioni contenenti i dati o le stringhe saranno visualizzate con il relativo indirizzo.
Nel caso siano presenti piu' locazioni di quante ne possa contenere lo schermo, i dati visualizzati scorreranno in alto.
Per terminare lo scrolling dello schermo sia per interrompere la funzione H sara' necessario premere il tasto di RUN/STOP. In questo caso resteremo in ambito VICMON.
Per far scorrere lentamente lo schermo e' sufficiente premere il tasto di controllo.

I ESEMPIO
Ammettiamo che il gruppo di dati $A9 2F 3C sia immagazzinato in memoria in una parte qualsiasi ma compresa fra gli indirizzi $C000 e $C0FF.
Per localizzarla con i relativi indirizzi opereremo come segue:

COMANDO: H C000,C0FF,A9,2F,3C (RETURN)

RISULTATO
Viene esaminata la memoria fra le locazioni assegnate e se il gruppo di dati richiesto e' presente viene visualizzato con l' indirizzo accanto.

II ESEMPIO
Vogliamo cercare la locazione esatta della parola COMMODORE che sappiamo essere presente fra le locazioni di indirizzo $2000 e $3000.

COMANDO: H 2000,3000,'COMMODORE (RETURN)

RISULTATO
Saranno visualizzate sullo schermo le locazioni di memoria ai cui indirizzi inizia la stringa richiesta.
Accanto a questi indirizzi verra' visualizzata la parola COMMODORE in reverse.

J = JUMP (cioe' salto ad una subroutine)

FORMATO: J

FUNZIONE: Esegue la chiamata ad una subroutine e ritorno senza un SINGLE-STEPPING mentre un programma sta girando sotto il comando W.

Il comando W, che esamineremo meglio in seguito, fa girare un programma una linea per volta o come si dice PASSO, PASSO.
Ad esempio dopo l'esecuzione di un comando presente in una linea, il programma attende un input da parte dell' operatore prima di procedere.
Usando il comando J e' invece possibile eseguire una subroutine tutta insieme cioe' non PASSO PASSO come il resto del programma.
La subroutine puo' essere stata testata in precedenza essa stessa e quindi sara' stata fatta magari girare con il modo W, ma ora non lo si reputa piu' necessario.

Questo comando quindi consente di muoversi attraverso le varie subroutines presenti nel programma molto rapidamente ed in maniera non dispersiva, soprattutto in relazione al tempo di esecuzione, ma tenendo d' occhio l' insieme delle

funzioni e la correlazione fra le stesse che il programma principale esegue.
Per ottenere questo risultato e' necessario che durante l' esecuzione del programma nel modo W il tasto J sia premuto quando sul video appare il comando di salto ad una determinata subroutine.

Ricordiamo inoltre che non e' necessario premere il RETURN e che il simbolo J relativo al comando JUMP non comparira' sullo schermo.

Quando e' usato il comando JUMP, cioe' prima di eseguire in modo rapido la subroutine, il corretto indirizzo di ritorno dalla routine stessa viene immagazzinato nello Stack Pointer.
In questo modo al momento in cui si arriva alla fine della subroutine e si incontra il comando di ritorno RTS, il Program Counter viene fissato all' indirizzo di ritorno nello Stack Pointer, dopo di che ritornera' in funzione il modo di esecuzione WALK.

ESEMPIO
Ipotizziamo di essere nel modo W e che sullo schermo compaiano i seguenti gruppi di istruzioni:

```
147F LDA £$00
1481 JSR $A2C7
```

COMANDO: J

RISULTATO
L' indirizzo $1484 che e' appunto quello della riga successiva alla 1481, viene inserito nello Stack Pointer.

Viene eseguita la routine con inizio a $A2C7, in modo normale ( cioe' non in modo W).
Al termine di questa routine il programma riparte dall' indirizzo il cui valore e' contenuto nello Stack Pointer.
Cioe' al rientro dalla routine il valore presente nello Stack Pointer viene ritrasferito al Program Counter.

L = LOAD (carica)

FORMATO: L " nome del file",(numero della periferica).

FUNZIONE: Carica in memoria il contenuto del file da una data periferica.

Il comando LOAD (L) consente,nello stesso modo del Basic, di caricare un file di dati o un programma da un determinata periferica nella memoria RAM del VIC.
Si possono quindi caricare files da disco o da cassetta.
Per i files disco, l' indirizzo della prima locazione RAM entro la quale il file dovra' essere letto deve essere costituita dai primi due Bytes del file.
I files provenienti da cassetta hanno come indirizzo di inizio parte dell' HEADER BLOCK iniziale.

ATTENZIONE

Con questo comando si possono caricare solo programmi o dati che siano stati precedentemente salvati su una periferica o con il comando S del VICMON o con il comando SAVE del BASIC del VIC, mentre non si possono caricare dati o programmi da cartridge.

Il comando e' composto dalla lettera L, dal nome del file e dal numero di periferica da cui deve essere letto.
Il nome del file deve essere racchiuso fra apici o virgolette (" ") e naturalmente si deve applicare la sintassi generale del Basic per questa operazione.
Ricordiamo che il numero di periferica per la cassetta e' 01 mentre quello del disco e' 08.
Quando viene usato il comando L, il file specificato fra virgolette e' letto fino a quando non si incontri un EOF (END OF FILE).
Nel caso non venga trovato il carattere di EOF ( o anche di EOT cioe' di END OF TAPE), e questo puo' accadere su ricerche da cassetta ed in particolare per file di dati, la funzione di LOAD non ha termine.
Anche questo classico esempio di LOOP infinito puo' essere arrestato con i tasti di RUN/STOP e di RESTORE.
Nel caso invece che il file non sia trovato sara' visualizzato il solito messaggio di errore ed il VIC sara' riportato in ambito Basic.

ESEMPIO
Ammettiamo di avere su disco un programma di nome TEST, che sia lungo 258 Bytes e che i primi due Bytes siano rispettivamente 00 e CA.
Si desidera caricare il file in memoria.

COMANDO: L"TEST",08 (RETURN)

RISULTATO
Il programma TEST presente e trovato sul dischetto e' caricato in memoria a partire dalla locazione $CAOO alla locazione $CBOO inclusa.

M = MEMORY

FORMATO: M (indirizzo)

oppure

M (indirizzo di partenza),(indirizzo di fine)

FUNZIONE: Visualizza i codici esadecimali contenuti in memoria.

Il comando M visualizza il contenuto della memoria dall' indirizzo di partenza specificato nel parametro all' indirizzo di fine incluso.

La visualizzazione mostrera' l' indirizzo in esadecimale ed il contenuto, sempre in esa, di 5 bytes di memoria.
Se invece di un blocco di memoria viene dato solo un indirizzo, allora saranno visualizzati i codici esadecimali (sempre 5 a riga) a partire da quell' indirizzo.
Gruppi di 5 bytes in piu' possono essere esaminati come al solito eseguendo lo scroll di schermo tramite l' uso dei tasti di controllo cursore.

Il contenuto della memoria puo' essere cambiato riscrivendo sopra al valore visualizzato e premendo successivamente il return.
L' indirizzo della prima locazione di memoria esaminata, relativamente al gruppo dei 5 bytes appare alla sinistra della riga.
Se si tenta di modificare i valori contenuti in zone di memoria riservate, come ad esempio i valori contenuti in ROM, allora accanto alla locazione di memoria immutabile apparira' un punto interrogativo (?) a segnalare l' impossibilita' della operazione.

ESEMPIO
Si desideri visualizzare i cinque bytes di memoria con indirizzo di partenza $1000 e variarne il contenuto.

COMANDO: M 1000 (RETURN)

SCHERMO: 1000 AO 00 EA EA FF

OPERAZIONE:Posizionare il cursore sopra il primo 0 della seconda locazione di memoria, cioe' quella che si trova a 00.
Digitare quindi FF e RETURN.

RISULTATO
I primi 5 Bytes di memoria con indirizzo alla locazione $1000 si leggono ora:

AO FF EA EA FF

N = NUMBER

FORMATO :N
(indirizzo),(indirizzo),(offset),(limite inferiore),(limite superiore),W

NB dove OFFSET e' un valore in esadecimale che indica l' ammontare che deve essere aggiunto all' indirizzo esistente.
Limite alto e basso specificano il range degli operandi che devono essere deviati.
W e' un parametro opzionale che sta ad indicare che si opera su una WORD TABLE.

FUNZIONE: Questo comando serve per riassegnare indirizzi assoluti in memoria, entro un determinato range, quando il programma e' stato riallocato con un comando T.

Con il comando T (per Transfer) si puo' rilocare il programma in un' altra parte della memoria.
Purtroppo se il vostro programma contiene degli indirizzi assoluti, questi non saranno piu' validi.
In questo caso il comando N vi consente di cambiare automaticamente questi valori.
Per operare e' necessario prima di tutto l' ammontare di memoria del programma o della subroutine da rilocare.
E' da notare che se avete spostato il programma ad una locazione di memoria piu' bassa, dovrete calcolare il cosi' detto valore WRAP-AROUND.

ESEMPIO:

Se il programma era allocato con indirizzo di partenza a $A000 ed e' stato spostato a $0400 ,quindi in una locazione piu' bassa, avete mosso $6400 fino a $A000+$6400=$10400.(NB. VALORE DI 5 BYTES)
Il valore $6400 e' il valore di offset.

Con il comando N potete cambiare tutti gli indirizzi assoluti o solo quelli entro un determinato range.
Il range e' stabilito fissando i limiti superiori e inferiori e ricordando che questi limiti sono inclusi.
Dovete anche specificare il blocco di memoria nel quale il cambiamento e' richiesto.
Il VICMON prendera' ogni operando entro il blocco selezionato e ci aggingera' l' ammontare che deve essere deviato (OFFSET).
Ad esempio controllerra' tre Bytes ed aggiungera' la deviazione ai successivi 2 bytes.
Se dovete cambiare una WORD TABLE, cio' potrebbe avere risultati disastrosi sull' intero programma, ma fortunatamente nel VICMON e' prevista l' opzione W al termine del comando N.
Quando infatti nel comando N e' presente l'opzione W, allora tutte le parole, es ogni 2 bytes, saranno deviate come e' stato descritto in precedenza.

ATTENZIONE

Non e' possibile usare il comando N nel range delle vostre locazioni DATA perche' sarebbero distrutti i DATA utilizzabili.

ESEMPIO: Ipotizziamo di aver utilizzato un comando TRANSFER per rilocare un programma compreso nelle

locazioni $1000 e $2000 alle locazioni $1500 fino a $2500.
Si sappia che sono presenti degli indirizzi assoluti e si desideri il loro corretto riutilizzo.

COMANDO: N 1500,2500,0500,1000,2000 (RETURN)

RISULTATO
Tutti gli indirizzi assoluti presenti nelle vecchie locazioni sono incrementati di 0500 nelle nuove locazioni.

Q = QUICK TRACE

FORMATO: Q

oppure

Q(indirizzo)

FUNZIONE: Far girare un programma a passo lento iniziando da un indirizzo specificato.

Il comando Q in modo simile a quanto avviene con l' uso del comando G, puo' essere usato con o senza un indirizzo.
Quando Q e' usato da solo, VICMON eseguira' il programma in memoria partendo dall' indirizzo in quel momento presente nel Program Counter.

NOTA

Si ricorda che per visualizzare il contenuto dei registri, fra i quali appunto il Program Counter, si puo' usare il comando R.

Se invece insieme al comando Q si fornisce anche un indirizzo, allora l' esecuzione del programma iniziera' dall' indirizzo specificato indipendentemente da cio' che contiene il Program Counter.

Come abbiamo detto all' inizio il comando Q funziona in maniera simile al comando G con una particolarita'.
Mentre G porta il controllo di programma completamente sotto la CPU, Q esegue una istruzione per volta, andando a vedere dopo ogni passo se non sia stato fissato un Breakpoint o se non avete chiesto di terminare l' esecuzione.
Questo controllo di Breakpoint vi consente di fissare dei punti di STOP in ROM altrettanto bene che in RAM.
Quando viene trovato un Breakpoint la esecuzione del programma si arrestera' e ci troveremo in modo WALK.

Per visualizzare i registri basta premere il RUN/STOP, digitare R seguito dal RETURN.
L' interruzione puo' essere generata in qualsiasi momento tramite la tastiera usando il RUN/STOP e successivamente il tasto X.
In questo caso verra' immediatamente fermata l' esecuzione del programma e il contenuto di tutti i registri ( in quel momento) visualizzata.

ESEMPIO
Si desideri eseguire un programma presente in memoria in modo QUICK TRACE iniziando dalla locazione $1000.

COMANDO: Q 1000 (RETURN)

RISULTATO
Il Program Counter e' fissato a $1000 e gli altri registri sono inizializzati.
Se abbiamo usato una pagina ZERO virtuale, questa viene immediatamente utilizzata al posto della pagina ZERO normale.
L' esecuzione del programma iniziera' dalla locazione $1000 e verra' portata avanti in modo QUICK TRACE.

R = REGISTER

FORMATO: R

FUNZIONE: Visualizza il contenuto dei registri.

Il comando R consente di vedere sullo schermo il contenuto dei seguenti registri del microprocessore 6502:

PC = Program Counter

SR = Status Register

AC = Accumulator

XR = Registro X

YR = Registro Y

SP = Stack Pointer

Questo comando puo' essere utile quando si sta provando un programma, perche' R vi consentira' di osservare se i registri contengono i valori che si desidera.

Si puo' cambiare il valore degli stessi registri quando ci si trova nel modo R, molto semplicemente riposizionandosi sopra i valori che appaiono sullo schermo, variandoli e premendo poi il RETURN.

ATTENZIONE

Quando si effettua un controllo dei registri e piu' ancora quando se ne cambia il contenuto di qualcuno sarebbe bene prendere nota scritta di quello che appare e di quello che si cambia.

I registri suddetti vengono automaticamente visualizzati:

-Quando entra in funzione VICMON

-Quando viene trovato un Breakpoint in conseguenza dell' esecuzione di un comando G.

-Quando durante l' esecuzione di un programma in

modo QUICK TRACE si termini premendo il tasto RUN/STOP in combinazione con il tasto X.

ESEMPIO
Visualizzare il contenuto dei registri

COMANDO:R (RETURN)

RISULTATO
Viene visualizzato il contenuto dei registri in questo formato:(I contenuti sono pero' ipotetici)

```
.R

      PC    SR  AC  XR  YR  SP
. ; 0401    33  00  63  00  F6
```

RB = REMOVE BREAKPOINT

FORMATO: RB(indirizzo)

FUNZIONE: Serve a rimuover i punti di fermata o BREAKPOINT.

I Breakpoint sono fissati con il comando B come visto in precedenza e possono essere rimossi con un comando di rimozione appunto REMOVE BREAKPOINT, semplicemente digitando le parole RB seguite dall' indirizzo della locazione di memoria dove era stato messo lo Stop.

Se nel punto dove si utilizza RB non esiste nessun breakpoint, allora VICMON interpretera' questo comando come un DISPLAY REGISTER e eseguira' le funzioni viste con il comando R.

ESEMPIO
Ammettiamo che sia stato fissato un BREAKPOINT alla locazione di memoria $1050 e che si desideri toglierlo.

COMANDO: RB 1050 (RETURN)

RISULTATO
Viene definitivamente tolto il Breakpoint alla locazione $1050.

S = SAVE

FORMATO: S"nome del file" ,(numero della periferica),(indirizzo),(indirizzo)

FUNZIONE: Scrive il contenuto di una data zona di memoria su una particolare periferica che potra' essere disco o cassetta.

Il comando S (SAVE) consente di salvare un programma o un gruppo di dati su cassetta o su disco per utilizzarli poi in un secondo tempo.

I parametri del comando S consistono nel nome del

FILE, nel numero della periferica (ricordiamo 01 per la cassetta e 08 per il disco) e negli indirizzi di inizio e fine dati che devono essere in questo caso specificati a differenza di quanto avviene per il simile comando SAVE del Basic.
Gli indirizzi di inizio e fine sono naturalmente indirizzi esadecimali della memoria RAM nella quale e' presente in quel momento il file da salvare.

Il nome del File deve essere racchiuso fra virgolette ("") e deve obbedire alle regole di sintassi dei comandi per la gestione dei files del VIC.
Ricordiamo quindi che per esempio deve iniziare con un carattere alfabetico e non deve essere piu' lungo di 16 caratteri.
L' indirizzo iniziale deve essere quello della locazione di memoria in cui incomincia il file, mentre quello finale deve essere di un Byte in piu'.

ATTENZIONE

- Se l' indirizzo finale non e' aumentato di un byte rispetto al reale, verra' perso l'ultimo carattere del file.

- Se la periferica specificata nel relativo parametro non e' presente sara' segnalato un messaggio di errore:

?DEVICE NON PRESENT ERROR

e torneremo in ambito Basic.

Con il VICMON non si puo' salvare memoria al di la' dell' indirizzo $7FFF.
L' indirizzo iniziale deve essere uguale o maggiore di $0000 ma non maggiore di $7FFF e l' indirizzo finale deve essere ovviamente piu' grande di $0000 ma non maggiore di $8000.
Se si tenta di salvare memoria al di fuori di questo range, sia su nastro che su disco viene scritto solo il FILE HEADER cioe' la testata e non il resto dei dati o dei programmi.

ESEMPIO
Ipotizziamo di avere un programma in memoria dalla locazione $1000 alla locazione $10FF e si desidera scrivere il programma su disco con il nome di TEST 1.

COMANDO: S "TEST 1",08,1000,1100 (RETURN)

RISULTATO
Il file programma denominato TEST 1 e' salvato su disco. Questo File conterra' il contenuto delle locazioni RAM dall' indirizzo $1000 a $10FF incluso.

T = TRANSFER

FORMATO: T(indirizzo),(indirizzo),(indirizzo)

FUNZIONE: Serve per trasferire i contenuti di un blocco di memoria RAM ad un' altra area di memoria.

Questo comando vi consente di rilocare un programma o dei dati in un' altra parte della memoria.
Questo puo' risultare utile qualora si desideri espandere un programma o se si vuole usare parti di un programma o di dati senza essere costretti a riscriverli.
Nel comando sono presenti 3 parametri relativi a tre diversi indirizzi.
I primi due delimitano il blocco di memoria che deve essere duplicato mentre il terzo indica l' indirizzo di inizio della copia.

Se il programma da trasferire contiene indirizzi assoluti o WORD TABLE, il trasferimento avverra' ma questi dati non avranno piu' una logica all' interno delle funzioni del programma.
Per questo si rimanda alle spiegazioni relative al comando N per una rilocazione accurata.

ESEMPIO
Ipotizziamo di avere un blocco di dati qualsiasi (programma, subroutines o data) in memoria dalla locazione $3000 alla locazione $3500 e che si vogliano avere ANCHE a partire dalla locazione $4000.

ATTENZIONE

La parola ANCHE usata nell' ultima riga sta a significare che si tratta di una vera e propria duplicazione e non di un trasferimento che nel senso stretto del termine lascerebbe la zona di memoria iniziale vuota.

COMANDO: T 3000,3500,4000 (RETURN)

RISULTATO
I dati sono ora presenti sia nelle locazioni di memoria da $3000 a $3500 che da $4000 a $4500.

W = WALK

FORMATO: W

oppure

W (indirizzo)

FUNZIONE: serve ad eseguire un programma una istruzione per volta.

Se e' usato da solo il comando W esegue l' istruzione alla linea indicata dall' indirizzo presente in quel momento nel Program Counter.
Se e' usato con un indirizzo allora eseguira' l' istruzione presente a quel dato indirizzo e continuera' con le modalita' che vedremo.
Quando si usa W la prima istruzione viene eseguita e la seconda visualizzata.
VICMON attendera' che sia premuta la barra spaziatrice per eseguire la seconda istruzione, visualizzera' la terza e cosi' via.
In questo modo non solo si riesce ad avere una esecuzione passo, passo ma si riesce anche a tenere sotto controllo l' intero programma mano a mano che procede nelle sue funzioni.

Per riportarsi dal modo WALK al modo normale, naturalmente sempre in ambito VICMON, basta premere il tasto di RUN/STOP.

Si puo' inoltre visualizzare il contenuto dei registri in qualsiasi momento.
Per far questo basta premere i tasto RUN/STOP, poi il tasto R seguito quindi dal RETURN.
Anche ogni subroutine girera' passo,passo a meno che non si voglia utilizzare il comando J.

ESEMPIO
Si voglia far girare un programma che inizia dalla locazione $1000 passo, passo.

COMANDO:W 1000 (RETURN)

RISULTATO
L' istruzione immagazzinata a $1000 e' eseguita e la successiva ( ATTENZIONE non necessariamente quella a $1001 o $1002) viene visualizzata.

AZIONE:premere la barra spaziatrice

RISULTATO
Viene eseguita la seconda istruzione e la terza e' visualizzata.

X = RITORNO AL BASIC

FORMATO: X

FUNZIONE: Serve per uscire dall' ambito VICMON e tornare in ambito Basic.

L' uso di questo comando vi riportera' in ambiente Basic.
Ricordiamo solo che l'eventuale programma in Linguaggio Macchina resta immagazinato in memoria, ma i relativi BREAKPOINT e l'eventuale pagina zero virtuale sono resettati.

ESEMPIO
Si desideri tornare la Basic

COMANDO: X(RETURN)

RISULTATO
Si rientra cosi' in ambito Basic e sara' visualizzata la scritta READY. con il cursore lampeggiante sotto

NOTE E PARTICOLARITA'

Inclusione di una routine in Linguaggio Macchina in un Programma Basic.

Il VICMON puo' essere molto utilmente impiegato per scrivere delle routines in Linguaggio Macchina piuttosto che per mettere a punto dei programmi completi per i quali consigliamo invece un ASSEMBLER piu' potente.
Dopo aver scritto la routine sara' quindi necessario inserirla nel corpo del programma principale, o MAIN PROGRAM, che sara' stato scritto in Basic.

Per far questo normalmente viene utilizzato il sistema di immettere la routines scritta in Linguaggio Macchina, sotto forma di comandi DATA.
Il procedimento e' il seguente.
All' inizio del programma Basic si fa eseguire un ciclo di POKE ( con un comando FOR...NEXT) e si immette la routine in Linguaggio Macchina o nel Buffer di cassetta o nella parte alta della memoria.

L' utilizzo pero' puro e semplice di questo metodo presenta alcuni inconvenienti. Vediamoli.

1 - La routine in Linguaggio Macchina prende 3 volte piu' spazio di memoria del necessario. Infatti per ogni Byte di Linguaggio Macchina sono necessari:

- Il Byte per il codice macchina

- Lo spazio necessario per il comando DATA.

- Le virgole per separare i vari codici all' interno del comando DATA.

- Il numero di linea

- I links fra le linee

- Il comando FOR...NEXT per immettere i codici nella scelta posizione di memoria.

2 - Non si possono utilizzare altre informazioni messe nei DATA, mentre in alcuni programmi e' necessario riutilizzare i contenuti dei DATA piu' volte, utilizzando il comando RESTORE.

E' quindi consigliabile utilizzare il VICMON per immettere la routine alla fine del programma Basic.
Cosi' facendo si eliminano i punti negativi visti in precedenza.
Si occupa infatti il minimo spazio indispensabile di memoria. Non sono necessari i comandi DATA con quello che ne consegue ed il programma Basic puo' essere tranquillamente ed in qualsiasi momento modificato.
C'e' solo una limitazione da ricordare.
La routine in Linguaggio Macchina deve essere rilocabile. In altre parole non deve contenere salti a posizioni fisse di memoria nel suo interno perche' il Basic e' in grado di rilocare la routine ma non di ricalcolare gli indirizzamenti interni.
In molti casi pero' non e' una limitazione seria .

Per spiegare quanto segue facciamo una breve disgressione dal contenuto di questo manuale e vediamo come opera il VIC nella memorizzazione dei programmi.
Il Sistema Operativo controlla il termine di un programma in due modi:

A) Considera la fine LOGICA del programma attraverso l' esame della linea di LINK, che in questo caso, cioe' alla fine di un programma, invece di contenere il prossimo indirizzo a cui saltare contiene 3 zeri.
Comunica cioe' al Basic quando fermare l' esecuzione di un comando LIST o RUN.

B) Considera e controlla anche la fine FISICA di un programma attraverso l' esame dei puntatori di indirizzo 45 e 46 che indicano il Byte immediatamente seguente la fine del programma stesso.
Questi puntatori sono utilizzati particolarmente per i comandi LOAD e SAVE e per rilocare parti di un programma Basic quando si effettuano dei cambiamenti come aggiunte o cancellazioni di righe di programma o di parti di esse.

Noi utilizzeremo questi puntatori per appendere la routine in Linguaggio Macchina al programma in Basic.
Ne cambieremo cioe' il valore per comunicare al Sistema Operativo che la lunghezza FISICA del programma e' stata aumentata.

Le seguenti linee di programma, inseribili preferibilmente al termine, serviranno per appendere una routine in Linguaggio Macchina ad un programma Basic anche nel caso siano state

inserite altre routines in Linguaggio Macchina

```
10000 I%=PEEK(46)*256 + PEEK(45)+4: PRINT%
10001 READJ%:IFJ% =0 THENPOKEI%,J%:I%=I% + 1:
GOTO10001
10002 PRINT I% : J%=I%/256 : I%=I%-J%*256:
POKE45,I%:POKE46,J% : CLR: END
```

NOTA

1- I DATA utilizzati per la routines in Linguaggio Macchina devono essere seguiti da un valore negativo che significhera' la fine del ciclo.

2- I comandi DATA usati per la routine in Linguaggio Macchina devono essere i primi o i SOLI DATA presenti nel programma.

3- Il "+4" nella linea 10000 rappresenta lo spazio necessario per le variabili I% e J%

4- Il comando CLR al termine della linea 10002 servira' a mettere a punto i due puntatori usati dal Basic per ricordare le variabili numeriche.

Non appena e' messa a posto la routine in Linguaggio Macchina si possono cancellare sia i comandi DATA utilizzati che la precedente routine.

Al termine di questo manuale e' riportato un breve capitolo relativo al linguaggio macchina oltre ad appandici di utilizzzo, programmi ecc.

# PROGRAMMER'S AID

Il PROGRAMMER'S AID e' stato messo a punto per aiutare sia il programmatore esperto che non a scrivere con piu' facilita' ed a far girare con maggiore rapidita' le procedure necessarie.
Questo risultato e' ottenuto con i comandi che entrano a farr parte del Sistema Operativo del VIC quando il Cartridge e' inserito.

## INSERIMENTO

Anche questo Cartridge deve essere inserito e rimosso a computer spento. Il punto di inserimento e' la porta di espansione. Ricordiamo che se stiamo usando una Memory Expansion Board con alimentazione separata, anche questa unita' deve essere spenta.
Inoltre, malgrado si affermi che questo Cartridge possa essere usato anche con il Monitor o con il Super Expander inseriti in una espansione qualsiasi vi consigliamo per esperienza diretta di disattivarli per evitare conflitti con il Sistema Operativo.
Per far entrare in funzione il Cartridge del PA, una volta correttamente inserito e riacceso il computer digitare:

SYS 28681
o
SYS 7x4096+9

(Return)

Sullo schermo sara' visualizzato:

------PROGRAMMER'S AID------

Ready.

A questo punto tutti i comandi del PROGRAMMER'S AID sono ora inseriti nel Sistema Operativo.

## I COMANDI DEL PROGRAMMER'S AID

### Introduzione e tasti funzione

In questa sezione sono descritti i modi del PROGRAMMER'S AID. Questo Cartridge consente un' esteso uso dei tasti funzione del VIC. Infatti, malgrado ci siano solo 4 tasti funzione nel VIC sono disponibili molte funzioni i cui valori possono essere assegnati ai tasti.
Premendo i tasti normalmente, si ottengono i valori relativi a F1,F3,F5 e F7 mentre per ottenere F2,F4,F6 e F8 e' necessario premere anche il tasto SHIFT. Premendo invece di SHIFT il tasto con il simbolo CTRL, avremo rispettivamente F9,F10,F11 e F12. Notare che questi valori non sono impressi sui tasti.

Come abbiamo detto il PROGRAMMER'S AID ha due modi di operare:

MODO PROGRAMMA

MODO EDIT

Ciascuno di questi due modi assegna differenti valori ai tasti funzione.
All' atto della messa in funzione del PROGRAMMER'S AID ci troviamo immediatamente in modo Programma. In questo modo di operare ai tasti funzione vengono assegnati i seguenti comandi:

KEY 1 = LIST
KEY 2 = MID$(

KEY 3 = RUN (Return)
KEY 4 = LEFT$(
KEY 5 = GOTO
KEY 6 = RIGHT$(
KEY 7 = INPUT
KEY 8 = CHR$(
KEY 9 = EDIT (Return)
KEY10 = GOSUB
KEY11 = RETURN
KEY12 = STR$(

MODO DI EDIT

Premendo CTRL e F1 insieme o digitando il comando diretto EDIT (Return) si passa appunto nel modo di EDIT.
In questo modo i tasti funzione assumono i seguenti valori:

KEY 1 = LIST
KEY 2 = AUTO
KEY 3 = RUN (Return)
KEY 4 = DELETE
KEY 5 = FIND
KEY 6 = CHANGE
KEY 7 = TRACE
KEY 8 = STEP
KEY 9 = PROG (Return)
KEY10 = RENUMBER
KEY11 = MERGE
KEY12 = OFF (Return)

Tutti i comandi possono essere usati purche' digitati per intero, sia quando ci troviamo in un modo che nell' altro. Il fatto di trovarsi in modo EDIT o in modo PROGRAM e' relativo SOLO all' uso

dei tasti funzione.

ATTENZIONE!!!

La linea di inizio di un comando e' indicata nelle seguenti spiegazioni con un (snl = START LINE NUMBER), mentre la linea di fine e' indicata con (fnl = FINISH LINE NUMBER).

AUTO

FORMATO: AUTO(snl),(intervallo fra le linee)
oppure
AUTO

FUNZIONE: Per incrementare e visualizzare numeri di linea di programma automaticamente.

Dopo aver messo in funzione il PROGRAMMER'S AID l' uso del comando AUTO senza specificare la linea d' inizio ne l' intervallo fra le linee, visualizzera' automaticamente la linea 100 con vicino il cursore.
Dopo aver inserito la linea di programma Basic ed aver premuto il Return, il sistema proporra' automaticamente la linea 110, poi la 120 e cosi' via.
Come specificato nel formato pero' si puo' adoperare questa funzione per specificare un indirizzo di partenza diverso da 100 ed un incremento diverso da 10.

ESEMPIO

Visualizzare numeri di linea programma automaticamente iniziando da 50 con un intervallo di 5.

Comando:AUTO 50,5

Video: 50

Digitare: PRINT (Return)

Video: 50 PRINT
       55

RISULTATO
Si incomincia'a scrivere da 50 con un incremento di 5.

RENUMBER

FORMATO: RENUMBER(snl),(intervallo fra le linee).

FUNZIONE: Rinumera automaticamente tutte le linee di programma compresi i salti GOTO e GOSUB.

RENUMBER cambiera' automaticamente tutti i numeri di linea nel vostro programma in modo tale che il listato incominci alla linea che avete definito e tutte le seguenti linee siano incrementate dell' intervallo definito nel parametro.

ESEMPIO
Si desideri rinumerare il seguente programma:

```
100 REM RENUMBER COMMAND
```

```
110 PRINT"BUONGIORNO"
120 FOR L= 1 TO 1000
121 NEXT
130 PRINT"(SHIFT/CLR-HOME)":GOTO 110
```

Comando: RENUMBER 200,3 (Return)

VISUALIZZA
Dopo il LIST:

```
200 REM RENUMBER COMMAND
203 PRINT"BUONGIORNO"
206 FOR L= 1 TO 1000
209 NEXT
209 PRINT"(SHIFT/CLR-HOME)":GOTO 203
```

DELETE

FORMATO: Delete(snl)-(fnl)

FUNZIONE: Cancella linee di programma dalla memoria del VIC.

Questo comando serve a cancellare linee di programma . Il suo funzionamento e' simile a quello relativo al comando LIST.
ESEMPIO
Cancella le prime due linee del seguente programma:

```
100 REM RENUMBER COMMAND
110 PRINT"BUONGIORNO"
120 FOR L= 1 TO 1000
121 NEXT
130 PRINT"(SHIFT/CLR-HOME)":GOTO 110
```

Comando: -110 (Return)

RISULTATO
Dando il LIST vi accorgerete che rimangono SOLO le linee 120,121 e 130.

FIND

FORMATO:FIND(codice Basic),(sln)-(fln)
o
FIND"(stringa"),(sln)-(fln)
o
FIND(carattere),(sln)-(fln)

FUNZIONE: Ricerca nel programma un dato comando o una stringa e visualizza la linea o le linee dove sono presenti

Anche questo comando, nelle sue tre versioni viste, invece di cercare per tutto il programma, puo' essere usato selettivamente cioe' per una ricerca solo su parti della procedura.
In pratica la ricerca parziale e quindi i valori da assegnare ai parametri sln e fln funzionano come per i parametri di LIST o di DELETE.
FIND ricerca un carattere o un codice Basic o una stringa in tutto o in parte del programma e visualizza tutte le linee che contengono quel carattere o quel comando eccetto quelle in cui il carattere o il comando e' messo fra virgolette.

ESEMPIO
Si desideri trovare il carattere C nel seguente

programma:

```
10REM COMANDO FIND
20PRINT"ABCDEFGH VERTICALE"
30A$="ABCDEFGH"
40FORC=1TO8
50PRINT MID$(A$,C,1)
60PRINT"----------"
70NEXT C
```

Comando: FIND C (Return)

RISULTATO
Ogni linea di programma che contiene C sara' visualizzata.

CHANGE

FORMATO:CHANGE(vecchio codice),(nuovo codice),(sln)-(fln)

oppure

CHANGE(vecchia stringa),(nuova stringa),(sln)-(fln)

FUNZIONE: Ricerca un codice Basic o una stringa di carateri e lo rimpiazza con un nuovo codice Basic o con una nuova stringa di caratteri.

CHANGE e' una utilissima funzione che consente di modificare rapidamente sopratutto le stringhe. Anche questo comando, nelle sue opzioni viste, funziona come nel LIST o DELETE e cioe' il cambio puo' avvenire solo su determinati gruppi di linee.

Le linee cambiate vengono visualizzate sullo schermo.
Notare che nei comandi REM il cui seguito non sia incluso fra virgolette il cambio non avviene. Per esempio, se noi abbiamo:

10 REM PRINT

con il comando:

CHANGE PRINT,PRINT,10-1000

non otterremo alcun effetto.

ESEMPIO
Si desideri cambiare la stringa di caratteri del programma visto in precedenza a proposito di FIND da:

"ABCDEFGH" in "12345678"

Comando:CHANGE"ABCDEFGH","12345678" (Return).

Video: 20PRINT"12345678 VERTICALE"
30A$="12345678"

RISULTATO
Tutti i caratteri stringa "ABCDEFGH" vengono cambiati in "12345678" ed ogni linea in cui e' stata effettuata la variazione e' visualizzata sullo schermo.

EDIT

FORMATO: EDIT

FUNZIONE: Serve per passare dal modo PROGRAMMA al modo EDIT.

Come abbiamo visto in precedenza esistono due modi nel PROGRAMMER'S AID : il modo PROGRAMMA ed il modo EDIT.
Per passare da un modo all' altro e' sufficiente premere CTRL e F1 oppure digitare EDIT o PROG.

KEY

FORMATO:KEY
o
KEY numero,"Comando"

FUNZIONE: Esegue la lista dei comandi assegnati ai vari tasti funzione e consente di cambiare il loro valore.

Questo comando vi consente di visualizzare le informazioni assegnate ai tasti funzione e di cambiarle secondo quanto desiderate.
I tasti funzione che rappresentano un vero aiuto alla programmazione possono essere programmati assegnando loro un comando Basic, un simbolo grafico, un numero, una stringa di caratteri o una combinazione di queste. La solo limitazione e' data dal fatto che non si possono superare i dieci caratteri.

NOTA. Per una trattazione approfondita dei tasti funzione si consiglia di leggere il manuale PERIFERICHE II della EVM.

ESEMPIO
Si desideri assegnare il comando PRINT al tasto

funzione F1.

Comando: KEY1,"PRINT" (Return)

RISULTATO
Si puo' ora visualizzare ed assegnare ad ogni linea di programma tutte le volte che si desidera il comando PRINT semplicemente premendo F1.

## HELP

FORMATO: HELP

FUNZIONE: Visualizza la linea sulla quale e' stato trovato un errore durante l' esecuzione di un programma e sottolinea l' inizio dell' errore in reverse

HELP opera solo se e' dato immediatamente dopo che un errore di sintassi e' stato scoperto dal Sistema Operativo del VIC e mentre il messaggio di errore e' visualizzato sullo schermo.

ESEMPIO
Trovare un errore nel seguente programma:

```
10 FOR CO=1TO10
20 PRINT CO+2*3.142
30 NEXT C
```

Digitare: RUN

Video: 7.284

```
?NEXT WITHOUT FOR ERROR IN LINE 30
Ready
```

Comando: HELP (Return)

Video: 30 NEXT C(con la lettera "C" in reverse)

## DUMP

FORMATO: DUMP

FUNZIONE: Visualiza i valori di tutte le variabili tranne quelle nelle matrici.

Le variabili saranno listate nello stesso ordine che hanno nel programma e saranno visualizzate nel formato:

VARIABILE = VALORE

Il valore di una variabile puo' essere variato riposizionandoci sopra il cursore, scrivendo il nuovo valore e rieseguendo il programma dal punto immediatamente successivo a quello in cui si assegnava il valore originario alla variabile.

ESEMPIO
Visualizzare il valore delle variabili contenute nel seguente programma:

```
10 A$="NUMERI CASUALI"
20 PRINT A$
30 X= INT(RND(8)*15)+1
40 Y= INT(RND(8)*7)+1
```

```
50 R =X*16+8+Y
60 POKE36869,R
70 FOR CO=1 TO1000:NEXT CO
80 GOTO 20
```

Azione: Dopo aver dato il RUN, attendere qualche secondo e premere il RUN/STOP.

Comando: DUMP

RISULTATO (video):

```
A$ = "NUMERI CASUALI"
X =5
Y =6
R =94
CO=995
```

## TRACE

FORMATO: TRACE

FUNZIONE: Durante l' esecuzione di un programma vengono visualizzate le linee del programma stesso mano a mano che vengono eseguite.

Il comando TRACE deve essere eseguito prima di far girare il programma. Apparira' sulla parte destra alta dello schermo del VIC una piccola finestra dove, precedute dal segno POUND, saranno visualizzate i numeri di linea di programma mentre lo stesso viene eseguito.
Il numero massimo di linee che effettueranno uno SCROLL-UP durante l' esecuzione di questo comando, sara' al massimo di 6.

La velocita' di visualizzazione di queste linee e' di circa 2 linee programma al secondo. Per rallentare la velocita' basta tenere premuto il tasto CTRL o lo SHIFT.
Per disabilitare il TRACE usare il comando OFF.
Naturalmente questi due comandi si usano in forma diretta.

STEP

FORMATO: STEP

FUNZIONE: Ferma il programma dopo ogni istruzione e visualizza la prima linea della prosssima istruzione.

Se viene eseguito il comando STEP prima di far girare il programma ogni istruzione sara' eseguita individualmente.
Anche in questo comando come abbiamo visto nel precedente, appare una finestra sulla parte destra in alto dello schermo che conterra' la linea di programma associata con quella istruzione e la successiva linea da eseguire.
Premendo il tasto SHIFT o CTRL verra' eseguita la successiva linea e visualizzata l' altra e cosi' via.
Per disabilitare lo STEP usare il comando OFF.

ESEMPIO
Eseguire un programma passo, passo.

Comando: STEP °Returné

RISULTATO
Dopo il RUN compare la finestra in alto con il

primo numero di riga che e' eseguito ed il programma si ferma.Premendo lo SHIFT si passa alla successiva istruzione.

## OFF

FORMATO: OFF

FUNZIONE:Disabilita i comandi TRACE e STEP.

OFF cancella i comandi TRACE e STEP. La finestra viene quindi cancellata dallo schermo e il programma continua la sua esecuzione a velocita' normale.

## PROG

FORMATO: PROG

FUNZIONE: Per passare dal modo EDIT al modo PROGRAM.

Questo modo da accesso ai tasti funzione con i comandi previsti e descritti in precedenza per questo.
Ricordiamo che il modo programma entra in esecuzione non appena si mette in funzione il PROGRAMMER'S AID.
Oltre questo ricordiamo anche che trovandosi in modo EDIT e' sufficiente premere CTRL e F1.

ESEMPIO
Si desideri entrare o ritornare in modo PROGRAMMA.

Comando:PROG °Returné

RISULTATO: Viene visualizzato:

PROG

Ready

MERGE

Formato:MERGE"(nome del programma)","(numero della periferica)"

Funzione: Carica da una periferica un programma preventivamente salvato e lo incorpora nel programma attualmente in memoria.

Ricordiamo che le periferiche, dalle quali si possono ricaricare programmi sono o la cassetta (numero di periferica 1) o il disco ( numero 8), ma che possono essere assegnati anche numeri diversi.
Se non viene specificato il numero della periferica questa verra' assunta come 1 cioe' la cassetta.
Se invece non viene specificato il nome del programma da unire verra' preso il primo che trova.

ATTENZIONE!!!

Prima di caricare il programma dalla periferica fare attenzione che non ci siano numeri di linea in comune con il programma in memoria. Per superare questo inconveniente comunissimo usare

preventivamente la funzione RENUMBER.

ESEMPIO
Unire il programma" ROUTINE SORT", su cassetta,con il programma presente in memoria.

Comando: MERGE"ROUTINE SORT",1 °Returné.
Azione: Dopo che viene visualizzato il:

PRESS PLAY ON TAPE

eseguire il comando con i tasti della cassetta. Al momento in cui, dopo la scritta:

LOADING ROUTINE SORT

appare il:

Ready.

i due programmi sono uniti entro la memoria del VIC.

KILL

FORMATO: KILL

FUNZIONE:Annulla la funzione del PROGRAMMER'S AID.

Quando viene eseguito il comando KILL le funzioni relative al PROGRAMMER'S AID vengono cancellate. Il VIC ritorna cioe' al funzionamento come quando il Cartridge non e' presente.
Notare che pero' le funzioni relative ai tasti F vengono mantenute inalterate.

ESEMPIO
Disabilitare il PROGRAMMER'S AID

Comando: KILL °Returné

RISULTATO
Quello descritto in precedenza

## FUNZIONI SPECIALI

Oltre a quanto abbiamo detto il PROGRAMMER'S AID ha 6 speciali funzioni di EDITING che possono essere usate premendo il tasto CTRL e una lettera come descritto di seguito:

CTRL A : Esegue lo scroll in alto del programma.

CTRL E : Cancella le parentesi nel modo insert.

CTRL L : Cancella i caratteri alla destra del cursore sulla linea di schermo.

CTRL N : Cancella dallo schermo tutti i caratteri nel programma dopo il cursore.

CTRL Q : Esegue lo scroll in basso di un listato.

CTRL U : Annulla le linee dello schermo sul quale e' posizionato il cursore.

L' uso di questi comandi e' implicito nella definizione.

NB. Se si usa una di queste speciali funzioni di EDIT per cancellare caratteri, questi saranno rimossi solo dallo schermo e non verranno cambiati sulle linee di programma.
Per cancellare linee o caratteri dalla memoria e quindi dal programma si deve usare la procedura standard del Basic

# HI-RESOLUTION GRAPHIC

## INTRODUZIONE

Il VIC 1211 HI-RES GRAPHIC CARTRIDGE e' un interessante e versatile aiuto alla programmazione.
Aggiunge al sistema operativo del VIC 20 una nuova gestione del colore, un gruppo di comandi grafici ed una serie di facilitazioni per comporre brani musicali.

Questo cartridge contiene anche un' aggiunta di memoria RAM di 3 K-Bytes che consente quindi di scrivere e far funzionare programmi di dimensioni maggiori.
Sono inoltre attivati i tasti di funzione programmabili che si trovano sulla destra del computer per facilitare e velocizzare la scrittura dei programmi.
Questi capitoli non sono stati scritti per insegnarvi a programmare in Basic, ma ne presuppongono invece se non una conoscenza molto approfondita, almeno un po' di pratica.

La parte relativa e' stata suddivisa in 6 sezioni:

1-Introduzione

2-Uso dei tasti funzione

3-La grafica ed i colori

4-La musica

# 5-Comandi di lettura e periferiche

MESSA IN FUNZIONE DEL HI-RES GRAPHIC

Per prima cosa ricordiamo di assicurarci che il computer sia spento (Power Off) quando si inserisce il HI-RES GRAPHIC CARTRIDGE.
Questi deve essere inserito nella porta di espansione del VIC con la SCRITTA rivolta in alto.
Se stiamo usando una VIC 1010 MEMORY EXPANSION BOARD o altro tipo di espansione con alimentatore separato ricordiamo di togliere tensione anche a questo.
Il HI-RES GRAPHIC puo' essere usato con le espansioni da 8 e 16 K-Bytes, mentre non deve essere impiegato con l' espansione da 3 K perche' questa memoria e' gia' presente nel cartridge.
Dopo aver correttamente inserito il cartridge e' sufficiente accendere il VIC. Sullo schermo apparira' la scritta:

```
* * * * CBM BASIC V2 * * * *

6519 BYTES FREE

READY.
```

E naturalmente il cursore lampeggiante sotto.
A questo punto tutti i comandi del HI-RES GRAPHIC, i nuovi colori, ed il nuovo set di caratteri grafici sono ora disponibili e connessi al Sistema Operativo del VIC.

I nuovi comandi disponibili, con qualche piccola eccezione che vedremo in seguito potranno essere usati nella stessa maniera degli altri comandi BASIC.
Inoltre ricordiamo che i 3 K Bytes di memoria

aggiunti, come del resto si vede dalla scritta che appare all' accensione, fanno parte integrante del cartridge e possono essere usati come una normale espansione.
Il HI-RES GRAPHIC assegna automaticamente 8 nuovi comandi ai tasti funzione presenti sulla destra della tastiera del VIC.

I comandi assegnati ai tasti funzione da F1 a F8, e che si attivano al premere del singolo tasto sono quelli relativi alla gestione grafica es.:

DRAW, CIRCLE, PAINT,ecc.

Per attivare la funzione corrispondente al tasto la cui scritta si trova nella parte inferiore del tasto stesso, cioe' gli F2,F4,F6,F8, oltre al tasto relativo premere lo SHIFT o il tasto con il simbolo della COMMODORE.

Vedremo in seguito come sara' possibile variare l' assegnazione dei comandi relativamente ai tasti funzione.

## COMANDI DEL HI-RES GRAPHIC

Diamo qui di seguito una lista dei comandi del HI-RES GRAPHIC che sono aggiunti al Sistema Operativo del VIC quando il cartridge e' inserito:

1-Comandi usati per la grafica:

GRAPHIC, COLOR, REGION, POINT, DRAW, CIRCLE, PAINT, CHAR, SCNCLR.

2-Comandi usati per la musica:

CTRL, SOUND, P, Q, V, S, O, T, R, C, D, E, F, G, A, B, £, $.

3-Funzioni di lettura valori:

RGR, RCOLR, RDOT, RSND, RPOT, RPEN, RJOY.

A questa lista va aggiunto il comando KEY che consente al programmatore di assegnare egli stesso valori o comandi ai tasti funzione.

## SCRITTURA DEI COMANDI

Tutti i comandi del HI-RES GRAPHIC che servono per creare i grafici possono essere usati sia in modo diretto che come parte di un programma tranne i comandi REGION e DRAW che invece devono essere

obbligatoriamente inseriti in una linea di programma.

Si ricordi che in modo diretto il comando che viene digitato, appare sullo schermo ed entra in esecuzione quando si preme il RETURN.

ATTENZIONE

In modo diretto i caratteri che seguono i comandi GRAPHIC1, GRAPHIC2, GRAPHIC3, non saranno visualizzati sullo schermo.
Sara' necessario quindi usare una grande attenzione nell' assicurarsi che i caratteri inseriti dopo questi comandi siano scritti correttamente.

I comandi per scrivere la musica possono essere usati sia in modo diretto che in modo programma.
In modo diretto il comando SOUND deve essere scritto e poi eseguito semplicemente premendo il RETURN.
I restanti comandi musicali sono eseguiti in modo diretto semplicemente premendo il tasto assegnato ad ognuno di essi.
Solo il comando KEY deve essere usato in modo diretto.

**** CONVENZIONI ****

Il formato di ogni comando del HI-RES GRAPHIC riportato in questo manuale segue le sottoindicate regole:

1-Le parole scritte in lettere maiuscole devono essere riportate esattamente come sono.Non e' quindi necessario usare il tasto SHIFT per ottenere le maiuscole.

2-Le parole scritte in minuscolo, sempre con riferimento a questo manuale, indicano uno o piu' parametri definibili dall' utente come ad esempio le coordinate dello schermo, il numero di un' ottava musicale, ecc.

3-Le parentesi tonde che si impiegano nei comandi di lettura funzione, DEVONO essere usate nello stesso modo in cui sono scritte.

4-Nel caso siano indicati altri simboli come virgola, punto e virgola, ecc.,anche questi vanno usati esattamente come sono scritti.

5-L' uso del tasto RETURN puo' essere indicato in vari modi.

6-E' importante ricordare che il VIC HI-RES GRAPHIC invece di usare la corretta parola inglese "COLOUR" usa la parola "COLOR" e NON riconosce che questa.

COMPATIBILITA' CON IL BASIC DEL VIC ED ALTRE NOTE

I comandi del HI-RES GRAPHIC seguono le stesse regole del Basic del VIC.
L' unica grande eccezione e' rappresentata dall' uso del codice THEN.
Se questo comando precede un comando tipico del HI-RES GRAPHIC allora prima di quest' ultimo devono essere inseriti i due punti (:).
Un programma scritto usando il HI-RES GRAPHIC non potra' girare senza che il cartridge del HI-RES GRAPHIC stesso non sia inserito nel VIC.
I programmi ed i dati possono invece essere salvati su periferiche o ricaricati da esse in modo normale.
Per fermare l' esecuzione di un programma mentre sta girando con i comandi del HI-RES GRAPHIC basta premere il tasto di RUN/STOP insieme al tasto di RESTORE.

SEGNALAZIONI DI ERRORE

Se un comando del HI-RES GRAPHIC viene inserito in maniera non corretta e' visualizzato un messaggio di :

?SYNTAX ERROR

E' sufficiente apportare la correzione ed il programma potra' riprendere a girare.

# LA GRAFICA

## Introduzione

Nella precedente sezione sono stati descritti i comandi forniti dal cartridge del HI-RES GRAPHIC per la gestione grafica.
I comandi consentono di tracciare punti,disegnare figure, far entrare parti di testo e colori sullo schermo in uno dei sedici colori senza accedere direttamente alle locazioni di memoria.
Questi comandi sono immessi come un qualsiasi comando Basic e possono essere usati direttamente o in modo programma.

## COORDINATE DELLO SCHERMO

Per la grafica lo schermo del VIC e' diviso in una matrice di 1024 x1024.
Una matrice deve essere vista come una griglia nella quale un particolare indirizzo e' fissato tramite le sue coordinate.
La coordinata e' un numero usato per stabilire la distanza di un punto su una griglia da una ORIGINE o punto 0.
Sono quindi necessari due numeri, uno che stara' ad indicare quanto e' distante il punto dall' origine in direzione orizzontale e l' altro la distanza ma in direzione verticale.
L' orizzontale e' chiamata asse X, la verticale asse Y.
La prima coordinata in tutti comandi grafici e' la coordinata X.

L' origine o punto 0 dello schermo nel HI-RES GRAPHIC e' situato nel punto piu' in alto a sinistra dello schermo stesso.
Le coordinate di un qualsiasi punto hanno un range variabile da 0 a 1024.

## REGISTRI DEL COLORE

All' accensione del VIC con il HI-RES GRAPHIC cartridge inserito, sono assegnati quattro registri di colore.
Ricordiamo che un registro e' semplicemente una speciale locazione di memoria usata per immagazzinare un valore.
Un numero variabile da 0 a 15 e' immagazzinato in ogni registro ed e' usato per indicare il colore.
I colori ed i valori associati ad essi, che quindi devono essere presenti nei registri sono:

0 BLACK
1 WHITE
2 RED
3 CYAN
4 PURPLE
5 GREEN
6 BLUE
7 YELLOW
8 ORANGE
9 LIGHT.ORANGE
10 PINK
11 LIGHT.CYAN
12 LIGHT.PURPLE
13 LIGHT.GREEN
14 LIGHT.BLUE

15 LIGHT.YELLOW

L' uso dei registri colore e' il seguente:

-Il valore presente nel registro 0 indica il colore usato all' interno dello schermo.

-Il registro 1 contiene il valore del colore usato per il bordo esterno dello schermo.

-Il registro 2 contiene il valore relativo al colore del carattere usato.Per esempio il colore del punto o della figura da tracciare.

-Il registro 3 contiene il valore del colore AUSILIARIO che puo' essere usato per disegnare figure.
Si puo' utilizzare questo registro solo in modo MULTI-COLOR o in modo MIXED come vedremo in seguito.

-All' accensione del computer, naturalmente con il HI-RES GRAPHIC in funzione i valori presenti nei registri sono i seguenti:

REGISTRO 0 = 1 (WHITE)

REGISTRO 1 = 3 (CYAN)

REGISTRO 2 = 6 (BLUE)

REGISTRO 3 = 0 (BLACK

## COMANDI GRAFICI

Tutti gli esempi dati in questa sezione sono riportati in modo programma.

### GRAPHIC

FORMATO: GRAPHIC n

FUNZIONE: serve a selezionare un appropriato set grafico dal Sistema Operativo del VIC .

Il comando GRAPHIC puo' essere usato in modo diretto o indiretto.
Ci sono 4 diversi set grafici utilizzabili.

GRAPHIC0

E' quello del VIC in modo normale.
Questo comando e' usato per riportare in visualizzazione normale la pagina di schermo precedentemente usata per la grafica.
E' usato comunemente alla fine dell' esecuzione di un programma grafico.

GRAPHIC1
Seleziona il set grafico MULTI-COLOR dal sistema operativo del VIC.
In questo modo lo schermo del VIC viene dimensionato per la grafica.
Ogni posizione sullo schermo occupa un' area di 16 punti di larghezza per 8 punti di altezza entro la

matrice di schermo di 1024 per 1024.
I punti inoltre possono essere disegnati sullo schermo usando uno qualsiasi dei registri di colore disponibili.
Tutti i colori del HI-RES GRAPHIC possono essere usati in questo modo di rappresentazione.

GRAPHIC2

Attiva il modo di alta risoluzione dei colori.
In questo modo ogni posizione sullo schermo occupa un' area di 8 punti di altezza per 8 di larghezza entro la matrice di schermo.
Questo modo di visualizzazione di punti sullo schermo e' due volte migliore di quella possibile con il precedente comando di GRAPHIC1.
Tuttavia in questo modi di risoluzione i colori nei quali i punti possono essere tracciati sullo schermo sono limitati a quelli contenuti nei registri 0 e 2, lo schermo e i colori del carattere.
Solo i colori da 0 a 7 possono cosi' essere usati.
Se verra' scelto un colore di valore maggiore di 7 nel comando COLOR, allora HI-RES GRAPHIC sottrara' 8 dal valore errato ed usera' il valore del corrispondente colore.

Es. se si usa 12, quindi un valore non accettabile HI-RES GRAPHIC eseguira':

13-8 = 5

ed utilizzera' il verde.

Il colore del carattere puo' essere cambiato

usando il comando REGION qualora si desideri usare piu' di un colore per disegnare i punti .

GRAPHIC3

Attiva sia il modo multicolore che l' alta risoluzione in dipendenza del valore contenuto nel registro di colore del carattere dopo che e' stato eseguito il comando COLOR.
Se il valore del colore rimane nel range da 0 a 7, tutti i disegni saranno visualizzati nel modo Alta Risoluzione.
Se il valore del colore e' invece piu' grande di 7 tutti i punti verranno disegnati sullo schermo nel modo MULTICOLORE.
Quindi questo comando e' molto utile quando si desideri usare sia il modo MULTICOLORE che l' ALTA RISOLUZION.

ESEMPIO:Si voglia selezionare , all' interno di un programma grafico il modo ALTA RISOLUZIONE

COMANDO: 100 GRAPHIC 2

RISULTATO: Viene attivato il modo di Alta Risoluzione entro il sistema operativo del VIC.

COLOR

FORMATO: COLOR sc,bo,cha,au

FUNZIONE:Serve per assegnare i valori relativi ai

colori ad ognuno dei 4 registri visti in precedenza.

Il comando COLOR serve per assegnare i colori per :

sc = SCHERMO

bo = BORDO

cha= CARATTERE

au = COLORE AUSILIARIO

Il colore dello schermo, selezionato opportunamente insieme al colore del bordo, puo' essere usato per cancellare punti dallo schermo che erano stati disegnati in colori differenti.
Il colore ausiliario e' usato molto spesso insieme al comando PAINT.
Il valore di ogni colore e' immagazzinato in uno dei quattro registri della memoria del VIC.
Solo i registri possono essere selezionati con il comando COLOR ed il numero del registro colore forma il primo parametro di tutti i comandi HI-RES GRAPHIC usati per disegnare grafici sullo schermo.

Quando e' attivato il sistema, con HI-RES GRAPHIC inserito, i quattro registri contengono rispettivamente i valori 1,3,6 e 0.
La disponibilita' dei registri colore per l' alta-risoluzione e per il modo multicolore e' la seguente:

Registro MULTI-COLORE

| | |
|---|---|
| 0 | Colore dello schermo |
| 1 | Colore del bordo |
| 2 | Colore carattere |
| 3 | Colore ausiliario |

Nell' alta risoluzione:

| Registro | ALTA-RISOLUZIONE |
|---|---|
| 0 | Colore dello schermo |
| 1 | NON DISPONIBILE |
| 2 | Colore carattere |
| 3 | NON DISPONIBILE |

Come possiamo vedere nel modo ALTA-RISOLUZIONE sono disponibili e quindi selezionabili solo 2 Registri colore, quello relativo allo schermo e quello del carattere.
Questa limitazione puo' essere aggirata cambiando il colore del carattere con il comando REGION nel caso sia necessario un diverso colore sullo schermo.

ESEMPIO
Si desideri selezionare uno schermo bianco, un bordo bleu, la scrittura del carattere in nero ed un colore ausiliario verde.

COMANDO: 110 COLOR 1,6,0,5

RISULTATO
Quando viene eseguita questa linea di programma viene visualizzato uno schermo bianco con un bordo bleu.
Tutti i grafici disegnati che usino il registro colore del carattere saranno neri.
Se viene usato il colore ausiliario, il disegno che ne deriva sara' verde.

POINT

FORMATO: POINT cr,x,y

oppure

POINT cr,x,y,x1,y1,....xn,yn

FUNZIONE: Serve a disegnare uno piu' punti sullo schermo in un dato colore (cr).

Nel modo multicolore, un punto puo' essere visualizzato sullo schermo in uno qualsiasi dei quattro colori disponibili nei quattro registri colore.
Nel modo Alta Risoluzione, i punti possono essere solo disegnati con quei colori disponibili nei registri relativi allo schermo o al carattere (registri 0 e 2).
Il comando, come vediamo nel Formato, puo' essere utilizzato per tracciare uno o piu' punti sullo

schermo e naturalmente x e y rappresentano le coordinate del punto o dei punti.

ESEMPIO
Disegnare un singolo punto sullo schermo nel colore nero.

COMANDO: 120 POINT 0,900,900

RISULTATO
Quando viene incontrato questo comando viene visualizzato un punto nero sulla meta' destra inferiore dello schermo cioe' di coordinate x e y =900)

ESEMPIO
Disegnare in verde 4 punti sullo schermo.

COMANDO: 130 POINT 5, 900, 100, 900, 300, 1000, 100, 1000, 300

RISULTATO
Nei punti delle relative coordinate vengono disegnati 4 punti verdi

REGION

FORMATO: REGION c

FUNZIONE: Cambia il colore del carattere.

Il comando REGION cambia il colore del carattere che era stato preventivamente assegnato con un comando COLOR.
Cambia cioe' il valore assegnato al registro 2 che e' appunto quello che definisce il colore del carattere.
Nel modo multicolore al parametro di questo comando puo' essere assegnato uno qualsiasi dei colori disponibili sul HI-RES GRAPHIC
Nel modo alta-risoluzione invece possono essere usati solo i colori da 0 a 7.

ESEMPIO
Cambiare il colore del carattere in rosso.

COMANDO: 140 REGION 2

RISULTATO
Quando viene eseguita questa linea di programma il valore 2 corrispondente al colore ROSSO e' inserita nel registro n. 2.
In seguito a questo comando quindi, tutti i punti tracciati sullo schermo ( o i caratteri) che usino il COLORE DEL CARATTERE saranno visualizzati in rosso fino al termine del programma o fino a quando il contenuto di questo registro non sara' cambiato da un' altro comando REGION.ln3
DRAW

FORMATO:DRAW cr,x,yTOx1,y1

oppure

```
DRAW cr,x,yTOx1,y1TOx2,y2...
```

oppure

```
DRAW cr, TOx,y
```

FUNZIONE:Disegna una retta fra due punti.

Il comando DRAW disegna una linea retta fra due punti dati nel colore indicato dal registro scelto.
Questo comando puo' essere usato in 3 modi:

1-Si puo' tirare (DRAW appunto) una linea tra due punti dello schermo specificando solo un gruppo di coordinate.
Si puo' usare ancora il comando per tracciare un' altra linea fra differenti punti sullo chermo.

2-Si possono tracciare piu' linee con un solo comando DRAW specificando piu' gruppi di coordinate separati dal codice TO.
Il primo gruppo di coordinate indica il punto d' inizio della prima retta che deve essere tracciata sullo schermo.
Il punto di termine di ogni linea diventa il punto d' inizio della linea successiva.

3-Si puo' usare DRAW specificando solo il punto di fine di una linea.
In questo caso il punto d' inizio della linea coincidera' con il punto di fine della ultima

figura grafica, cerchio o punto singolo che sia.
In quest' ultimo caso se nessuna figura e' tracciata sullo schermo e quindi non ci sono per il comando dei punti di riferimento, il punto d' inizio di questa linea sara' dato dai valori di coordinate 0,0, cioe' dalla origine della matrice disponibile sul HI-RES GRAPHIC

ESEMPIO
Si desideri disegnare un rettangolo.

COMANDO: 150DRAW2,100,100 TO 200,100 TO 200,100 TO 200,200 TO 100,200 TO 100,100.

RISULTATO
Quando viene eseguita questa linea di programma un piccolo rettangolo rosso e' disegnato sulla sinistra in alto dello schermo.

ESEMPIO
Disegnare una linea diagonale al centro dello schermo.

COMANDO: 160 DRAW 2,TO 512,512

RISULTATO
Ammettendo che questa linea di programma sia eseguita successivamente a quella dell' esempio precedente, allora verra' tracciata una linea rossa dal punto in cui il VIC ha terminato di disegnare il rettangolo al centro ( ricordando che la griglia o matrice e' 1024 x 1024 allora 1024:2=512) dello schermo.

## CIRCLE

FORMATO: CIRCLE cr,x,y,rx,ry

oppure

CIRCLE cr,x,y,rx,ry,as,af

FUNZIONE: Disegna una figura circolare sullo schermo.

Le coordinate presenti nel comando CIRCLE definiscono la locazione di schermo del centro di un cerchio o di una figura circolare.
Il successivo gruppo di parametri danno la larghezza e l' altezza di questa figura circolare dal suo centro.
Percio' quest' ultimo insieme di coordinate vi consente di tracciare figure circolari schiacciate come per esempio un' ellissi.
Poiche' lo schermo e' rettangolare, malgrado le assi x e y abbiano lo stesso numero di divisioni, per avere un cerchio perfetto il valore di rx non potra' essere uguale a quello di ry.
Per disegnare un cerchio perfetto sara' quindi necessario moltiplicare il parametro rx per 0.7.

Questo comando puo' essere utilizzato anche per disegnare un' arco di cerchio. Per questo si deve aggiungere un' altro gruppo di parametri al CIRCLE.
Il primo parametro (as) e' il punto d' inizio dell' arco sulla circonferenza del cerchio ed il secondo parametro e' il punto di fine dell' arco.
I valori usati in questi parametri non sono espressi in gradi ma in GRADIENTI.

Un angolo giro (cioe' 360 gradi) e' composto da 100 gradienti.

ESEMPIO
Disegnare un cerchio.

COMANDO:165 CIRCLE 2,512,512,70,100

RISULTATO
Quando viene eseguita questa linea di programma, un cerchio rosso sara' disegnato nel centro dello schermo.

ESEMPIO:disegnare due archi di cerchio.

COMANDO:170 CIRCLE 2,800,300,140,100,40,60.

180 CIRCLE 2,800,300,140,100,40,60.

RISULTATO:Quando vengono incontrate queste due linee sullo schermo sono disegnate sullo schermo una coppia di parentesi.

PAINT

FORMATO: PAINT cr,x,y

FUNZIONE: Riempe un' area definita con un colore.

Il comando PAINT riempe un'intera area con il colore presente nel registro selezionato.
L' area deve essere completamente chiusa perche' in caso contrario il colore scelto con il PAINT riempira' l' intero schermo.
L' area da dipingere e' specificata da un qualsiasi punto all' interno dei suoi limiti.
Ogni area puo' essere riempita una sola volta.
Se si sta usando il set d' istruzioni del GRAPHIC 3 l' area che e' stata tracciata in modo MULTICOLOR dovra' essere riempita con lo stesso colore usato per disegnarla.
Cioe' limitatamente all' uso dell' istruzione GRAPHIC 3 e nel modo multicolre, non si puo' disegnare un rettangolo verde e pretendere di riempire l' area racchiusa dal rettangolo in rosso.

ESEMPIO
Si disegni un cerchio in ROSSO e lo si riempia in GIALLO.

```
COMANDO:190 CIRCLE 2,300,800,70,100

        195 REGION 7
        200 PAINT 2,300,800
```

RISULTATO
Quando viene eseguita questa parte di programma, nella parte a sinistra in alto dello schermo apparira' prima un cerchio rosso che sara' successivamente riempito al suo interno di giallo.

CHAR-TEXT DISPLAY

FORMATO: CHAR riga,colonna,"STRINGA"

FUNZIONE: serve a visualizzare un normale testo entro lo schermo selezionato per la grafica.

Il comando CHAR serve per visualizzare una normale stringa e quindi usata come testo, ma su schermo grafico senza dover ridisegnare i caratteri.
I parametri definiscono la riga e la colonna di partenza.
Questo comando puo' essere usato nel modo GRAPHIC 2 e nel modo GRAPHIC 3 quando il valore contenuto nel registro colore del carattere e' minore di 8.
Non e' invece consentito usare CHAR quando e' stato selezionato il GRAPHIC 1 multicolore.

ESEMPIO
Visualizzare la parola cerchio in nero in un testo grafico.

COMANDO: 210 REGION 0

220 CHAR 5,15,"CERCHIO"

RISULTATO
La parola CERCHIO apparira' in alto a destra sullo schermo.

SCNLR

FORMATO: SCNRL

FUNZIONE: Pulisce lo schermo grafico.

Il comando SCNRL e' usato in modo grafico per pulire lo schermo nello stasso modo in cui nel BASIC si usa il comando:

PRINT"(SHIFT CLR/HOME)"

Questo comando puo' essere usato sia in modo diretto che in modo programma.
Si suggerisce, in particolare quando si stanno provando dei nuovi programmi grafici di assegnare questo comando ad un tasto funzione.

NOTA.
Nell' implementazione di comandi grafici e soprattutto di routines grafiche e' consigliabile, in particolare quando si debbano soprammettere piu' risultati di utilizzare dei cicli di temporizzazione del tipo:

```
1000 FOR I =1 TO 2500:NEXT
```

che serviranno a visualizzare meglio i singoli passi del programma.

# LA MUSICA CON IL HI-RES GRAPHIC

## INTRODUZIONE

Questa sezione riporta tutti i passi necessari per suonare , intesa qui la parola come nel senso di PRODURRE SUONI, cioe' musica o rumori sul VIC usando il HI-RES GRAPHIC cartridge.
Sia la musica che i suoni o rumori possono essere messi in funzione sia direttamente che in modo programma.
Si puo' suonare una serie di note allo stesso tempo, per esempio un accordo musicale, o una stringa di note.
Entro un programma, una frase musicale che usa una singola nota e' creata inserendo un carattere del HI-RES GRAPHIC entro un comando Basic PRINT e quindi "STAMPANDO" la musica.

La musica puo' anche essere creata in modo diretto, senza usare il comando PRINT semplicemente premendo i tasti del VIC dopo essere entrati nel modo musicale.
Per creare un accordo il comando SOUND e' usato sia in modo diretto che in modo programma ed e' anche usato per generare un rumore.

## ATTENZIONE

I picchi delle note o dei rumori possono differire da macchina a macchina, per cui e' probabile che un' orecchio esperto trovi differenze di tonalita' fra lo stesso programma suonato su due diversi VIC.

## REGISTRI SONORI

Quando viene acceso il VIC con il HI-RES GRAPHIC inserito, sono inizializzati 5 registri.
Il registro e' una particolare locazione di memoria che immagazzina un valore.
I registri sonori del VIC, i valori dei quali sono immessi ai loro posti dopo l' esecuzione del comando SOUND (vedi dopo) sono 5.
I registri sonori da 1 a 4 si riferiscono alle 4 VOCI del VIC mentre il registro 5 e' relativo all' intensita' del suono.
Un valore da 128 a 255 contenuto in uno dei primi 4 registri causera' un' emissione sonora che variera' in funzione del registro o VOCE al quale il valore stesso e' associato.
Se il valore e' inferiore a 128 non ne' sortira' alcun effetto sonoro.

I registri da 1 a 3 generano note musicali ed ognuno di loro ha nel suo range 3 ottave complete.
Questi 3 registri possono suddividersi, come tipo di emissione sonora in:

BASSO

SOPRANO

ALTO

Ogni registro e' di un' ottava piu' alto di quello che lo precede.
Ad esempio una nota suonata nella III ottava della VOCE ( o registro ) 1 ha lo stesso valore musicale dell' identica nota ma suonata nella II ottava della voce 2 e della I ottava della voce 3.

Il registro 4 e' usato per creare il cosi' detto " RUMORE BIANCO" o WHITE NOISE per effetti speciali.

Il registro 5 contiene il valore del livello di volume sia delle note musicali che del rumore.
Il valore contenuto in questo registro deve essere compreso fra 0 (off) e 15 (massimo livello di volume).

## COMANDI MUSICALI

In questa parte viene mostrato il formato di ogni comando musicale attivabile con il HI-RES GRAPHIC, la sua funzione ed il suo uso evidenziandolo con degli esempi.

### SOUND

FORMATO: SOUND s1,s2,s3,s4,s5

FUNZIONE: Crea effetti sonori multivoce

Il comando SOUND consente che accordi musicali possano essere suonati facilmente sul VIC.
I parametri da s1 a s3 definiscono il valore delle note musicali ognuna nel range da 128 a 255 che devono essere suonate in uno dei tre registri musicali del VIC.
Un valore inferiore a 128 per ognuno di questi parametri escludera' la possibilita' di effettuare i suoni associati alla voce relativa ( per i suoni vedere anche l' appendice F del vostro VIC USER GUIDE).
Il parametro s4 definisce il valore del cosi' detto "SUONO BIANCO" usato per creare effetti sonori speciali. Anche in questo caso il valore di questo registro deve essere compreso fra 128 e 256.
Il parametro s5 definisce il livello del volume che va da 0 (OFF) a 15 (MASSIMO).
Il comando SOUND e' disattivabile assegnando il valore 0 a tutte le voci o assegnando il valore 0

al parametro s5.

ESEMPIO
Si desideri una serie di accordi uniti.

Programma:

```
300 FORX=1TO50:SOUND225,225,225,0,9
310 NEXT X
320 FORX=1TO50:SOUND230,230,230,0,9
330 NEXT X
340 FOR X=1TO50:SOUND235,235,235,0,9
350 NEXT X
360 SOUND0,0,0,0,0
```

RISULTATO
Quando questa subroutine va in esecuzione viene suonato un triplice accordo musicale.

INGRESSO MODO MUSICALE

FORMATO: CTRL e freccia a sinistra

FUNZIONE: Serve per entrare nel modo musicale gestito dal S.E.

Questo comando puo' essere usato sia in modo diretto che indiretto per consentire di utilizzare il modo musicale del S.E.
Il comando viene eseguito premendo il tasto CTRL e la freccia a sinistra.
Si puo' usare il comando fra virgolette per "STAMPARE" (PRINT) una serie di note musicali e

suonare il brano composto.
Quando si usa questo comando con il PRINT, viene visualizzata una F in campo inverso. In modo diretto si deve usare questo comando fuori dalle virgolette e nessun simbolo sara' visualizzato sullo schermo.

Si devono usare allora i tasti del VIC per suonare le singole note come se fosse la tastiera di un organo o di un pianoforte. Per terminare il modo musicale premere il tasto Return.
Quando si entra nel modo musicale con CRTL e freccia a sinistra, il S.E. interpretera' solo i caratteri che seguono ignorando gli altri nel caso vengono premuti. I caratteri musicali sono:

P Q V S O T R C D E F G A B £ $

Entro un programma si possono inserire fino a 72 di questi caratteri in ciascun comando PRINT, cioe' fra il simbolo descritto e la virgoletta di chiusura.
Qualora si desideri che la successiva stringa di caratteri sia suonata immediatamente dopo, cioe' in sequenza, e' necessario per ogni stringa non superare i 71 caratteri e farla seguire da un punto e virgola. Lo stesso concetto di stampa e di visualizzazione che si adopera in un normale programma Basic.
Per fermare l' esecuzione di un brano musicale basta premere RUN/STOP e RESTORE.

ESEMPIO
Si desideri scrivere della musica in modo programma.

Programma:

10 PRINT "(CTRL e freccia a sinistra)....."

RISULTATO
Viene visualizzata una F in campo inverso. Qualsiasi carattere, di quelli segnalati in precedenza, che venga inserito fra le virgolette al posto dei puntini, sara' interpretato coma una nota e sara' suonato.

ESEMPIO II
Si desideri iniziare a suonare la musica in modo diretto.

Azione:Premere il tasto CTRL e freccetta a sinistra

RISULTATO
In questo modo e' stata abilitata la tastiera del VIC,limitatamente ai tasti menzionati, a suonarre la musica come fosse un organo.

P

FORMATO: P

FUNZIONE: Per visualizzare tutti caratteri musicali sullo schermo.

Il comando P che puo' utilizzarsi in modo diretto o indiretto e' usato per visualizzare i caratteri

che sono stati adoperati per creare brani musicali nello stesso tempo che il brano stesso va in esecuzione.

ESEMPIO
Visualizzare i caratteri musicali in modo programma.

Programma:

```
10 PRINT"(CTRL/freccia a sinistra)P"
```

RISULTATO
Quando questa linea di programma viene eseguita allora tutti i caratteri musicali che seguono saranno visualizzati mentre il resto del programma gira.

## Q

FORMATO: Q

FUNZIONE: Cancella la visualizzazione dei caratteri musicali.

Il comando Q che puo' essere usato sia in modo diretto che indiretto arrestera' la visualizzazione dei caratteri durante il modo musicale.

ESEMPIO

Si desidera suonare con la tastiera in modo diretto senza che i caratteri adoperati siano visualizzati sullo schermo.

Azione:Premere CTRL e freccia a sinistra per entrare nel modo musicale del S.E. in forma diretta. Successivamente premere Q, poi C e D.

RISULTATO
Saranno suonate le note musicali corrispondenti ai tasti C e D senza che queste due lettere siano visualizzate.

V

FORMATO:V n

FUNZIONE: Fissa il volume della musica.

Il comando V fissa il volume che sara' usato per suonare le singole note con il S.E. e per esempio non e' usato insieme al comando SOUND.
Il valore del parametro n usato in questo comando puo' andare da 0 (off) a 9(il piu' alto).

ESEMPIO
Fissare il massimo volume prima di incominciare a scrivere un programma.

Programma:

```
300 PRINT"(CTRL/freccia a sinistra)PV9"
```

RISULTATO
Quando questa linea di programma e' eseguita, tutta la musica successiva verra' suonata al piu' alto livello di volume. Questo livello di volume verra' mantenuto fino a quando non si incontri un comando di cambiamento.

S

FORMATO: S n

FUNZIONE:Seleziona il modo musicale

Il comando S e' usato per selezionare una delle 4 voci o registri disponibili con S.E. per suonare la musica.
Ricordiamo che la scelta della voce determina il PICCO della nota o la sua altezza. Ogni voce dispone di 3 ottave. Una ottava e' appunto un insieme di 8 note.
I registri sono numerati da 1 a 4. I primi 3 selezionano rispettivamente il Basso, Medio e alto, mentre il IV e' usato per creare effetti sonori detti "RUMORE BIANCO".

ESEMPIO
Si desideri selezionare il registro 3

Programma:

300 PRINT"(CTRL/freccia a sinistra) PV9S2"

RISULTATO
Tutta la musica che viene suonata dopo questa istruzione sara' eseguita al volume 9 e con tono alto (registro 3) fino al successivo comando di cambiamento.

O

FORMATO: O n

FUNZIONE:Seleziona un' ottava musicale

Il comando O (attenzione a non scambiarlo con lo O zero) serve per selezionare un gruppo di otto note.
Ogni registro musicale dispone di 3 gruppi di otto note.
Ogni registro inizia un' ottava piu' alta di quella che lo precede.
Per esempio una voce suonata nella terza ottava del registro 1 ha lo stesso valore della stessa nota suonata nella seconda ottava della voce 2 e nella prima pttava della voce 1.

ESEMPIO
Selezionare la terza ottava nella precedente linea di programma.

Programma:

```
300 PRINT"(CTRL/freccia)PV9S203...."
```

RISULTATO
Quando questa linea di programma viene eseguita, tutte le note dopo il numero3 e la fine delle virgolette ( in pratica quelle che sostituiscono i puntini) saranno suonate nella terza ottava della seconda voce.

T

FORMATO: T n

FUNZIONE: Seleziona la durata di una nota.

Il comando T seleziona la durata o il tempo di una nota secondo i valori riportati nella seguente tabella:

| TEMPO | BEATS | DURATA |
|---|---|---|
| 0 | 900 | 4 |
| 1 | 600 | 6 |
| 2 | 450 | 8 |
| 3 | 300 | 12 |
| 4 | 225 | 16 |
| 5 | 150 | 24 |
| 6 | 112.5 | 32 |
| 7 | 56.25 | 64 |
| 8 | 28 | 128 |
| 9 | 14 | 255 |

NB La durata e' in sessantesimi di secondo.

ESEMPI
Selezionare il tempo nella linea di programma precedente.

Programma:

```
300 PRINT"(CTRL/freccia)PV9S203T6.."
```

RISULTATO
Le note fra T6 e la fine delle virgolette saranno suonate per la durata di 32/60esimi di secondo.

C D E F G A B

FORMATO: le lettere viste

FUNZIONE: Suonare la musica

Le lettere viste servono per suonare la musica sia in modo diretto che indiretto, da sole o in comabinazione fra di loro.
In modo diretto le note sono suonate all' atto della pressione del tasto corrispondente, mentre in modo indiretto queste sono incluse fra parentesi come parte di un comando PRINT.
In tutti e due i modi non e' comunque necessario separare i caratteri.

ESEMPIO

Suonare un accordo utilizzando la precedente linea di programma.

Programma:

```
300 PRINT"(CTRL/freccia)PV9S203CDE"
```

RISULTATO
Quando viene eseguita questa linea di programma sara' suonato un accordo musicale composto dalle tre note specificate i cui simboli saranno anche visualizzati.

## R

FORMATO:R

FUNZIONE:Pausa musicale

R e' usato per creare un periodo di silenzio o pausa in una composizione musicale. La lunghezza della pausa e' determinata dalla scelta del tempo effettuata con il comando T.

ESEMPIO
Inserire due pause nel programma.

Programma:

```
300 PRINT"(CTRL/freccia)PV9S203T6CRDRE"
```

RISULTATO

Quando suoneremo questa musica sara' creato un tempo di attesa fra le singole note , tempo pari a T6.

£ e $

FORMATO:£, $

FUNZIONE: esegue acuti e bemolle

Questi comandi sono usati come i pedali del pianoforte.
Il simbolo £ (POUND) serve a dare un tono piu' acuto alla nota che lo segue, mentre il $ fara' si che la nota seguente sia sonata in bemolle.
Questi comandi hanno effetto solo sulla sola nota seguente alla loro imputazione.

## COMANDI DI LETTURA VALORI

### Introduzione

Questa sezione illustra le sette funzioni di lettura che fanno parte del gruppo di comandi resi disponibili dal HI-RES GRAPHIC
Queste funzioni possono essere usate in modo diretto ed in modo programma. In modo diretto la FUNZIONE deve essere preceduta dal 'codice Basic PRINT seguito naturalmente dal Return.
Esse consentono di determinare e visualizzare i valori creati da alcuni comandi deisponibili sul HI-RES GRAPHIC o sulle periferiche per il controllo giochi.
Per esempio, usando una di queste funzioni si possono leggere i valori dei registri colore che saranno stati fissati con il comando COLOR o i registri sonori fissati con il comando SOUND.
Se nel programma si usano le PADDLES, la JOYSTICK o la LIGHT PEN, si puo' usare una FUNZIONE per determinare il valore di queste speciali periferiche.

### RGR

FORMATO: RGR

FUNZIONE: Legge il tipo di grafica scelta con il comando GRAPHIC.

Questa FUNZIONE legge il numero del modo grafico che era stato fissato con il comando GRAPHIC. Ogni numero da 0 a 255 puo' essere usato come parametro di questa FUNZIONE.
Infatti mentre un parametro va comunque fissato, il valore assegnato non ha nessun effetto sul valore che sara' restituito dalla FUNZIONE.
RGR e' particolarmente utile in modo programma quando si desideri passare da un modo grafico ad un' altro e poi riportarsi all' originale. Non e' invece utilizzabile in modo diretto perche' il valore restituito sarebbe 0.

ESEMPIO
Disegnare 3 cerchi cambiando colore e modo grafico.

Programma:

```
10 GRAPHIC 2
20 COLOR1,6,0,10
30 CIRCLE2,512,512,70,100
40 X=RGR(0)
50 GRAPHIC 3
60 REGION 10
70 CIRCLE3,850,750,70,100
80 GRAPHIC X
90 REGION 0
100 CIRCLE2,300,800,70,100
110 FOR Z=1TO1000:NEXT:GRAPHICO
```

RISULTATO
Dopo che e' stato dato il RUN a questo programma, viene assegnato il numero del primo modo grafico alla variabile X e questa variabile e' fissata sul III modo grafico.

## RCOLOR

FORMATO: RCOLOR(cr)

FUNZIONE: legge il colore contenuto in uno degli appositi registri.

Questa FUNZIONE leggera' il valore del colore contenuto nel REGISTRO COLORE che e' stato selezionato in precedenza con il comando COLOR o cambiato successivamente con il comando REGION.
La FUNZIONE di questo comando e' specialmente utile nel modo ALTA RISOLUZIONE quandi si desidera disgnare un punto sullo schermo con un colore diverso dal colore dello schermo stesso o dal colore del carattere e non si possa o non risulti conveniente utilizzare il comando REGION.

ESEMPIO
Si voglia disegnare un cerchio in alta risoluzione usando il colore carattere e riempiendolo con un colore ausiliario.

Programma:

```
10 GRAPHIC 2
20 COLOR1,6,0,2
30 CIRCLE 2,512,512,70,100
40 Q=RCOLOR(3)
50 REGION Q
60 PAINT 2,512,512
```

RISULTATO
Il colore contenuto nel registro colore numero 3

sara' letto ed usato per riempire il cerchio disegnato.

RDOT

FORMATO:RDOT(x,y)

FUNZIONE:Per ottenere il valore del colore di un punto presente sullo schermo.

La FUNZIONE RDOT leggera' il valore del colore di un punto sullo schermo nella posizione indicata dai parametri x e y indicati nella FUNZIONE.
Questa FUNZIONE e' utilissima nel modo multicolore quando si desideri disegnare parti o segni grafici sullo schermo con colori di riempimento.

ESEMPIO
Si desideri disegnare e riempire un cerchio in modo multicolore e disegnare un secondo cerchio con il colore usato per riempire il primo.

Programma:

```
10 GRAPHIC 1
20 COLOR 1,0,0,10
30 CIRCLE 2,512,512,70,100
40 PAINT 3,512,512
50 X=RDOT(512,512)
60 REGIONX:CIRCLE2,750,750,70,100
```

RISULTATO

Viene disegnato un cerchio nero, riempito in rosa e disegnato un secondo cerchio rosa.

RSND

FORMATO:RSND(n)

FUNZIONE:Legge il valore di un registro sonoro.

La FUNZIONE RSND legge il valore di una nota in uno dei primo 4 registri fissati con il HI-RES GRAPHIC ed il valore del volume fissato con il V registro sonoro.
Questa FUNZIONE puo' essere usata solamente per determinare il contenuto dei registri sonori che seguono il comando SOUND.

ESEMPIO
Usare la RSND entro un programma.

Programma:

```
10 X=225:SOUNDX,X,X,0,9
20 Y=RSND(5)
30 FORZ=YTOOSTEP-1
40 SOUNDX,X,X,0,Z
50 FORC=1TO500:NEXTC,Z
60 FORZ=OTOY
70 SOUNDX,X,X,0,Z
80 FORC=1TO500:NEXTC,Z
90 SOUNDX,X,X,0,0
```

RISULTATO
Verra' suonato un programma prima a volume cadente e poi ascendente.

RPOT

FORMATO:RPOT(n)

FUNZIONE:Legge il valore di una PADDLE.

La FUNZIONE RPOT leggera' il valore di una PADDLE e riportera' un valore compreso fra 0 e 255.
Cio' rappresenta il valore della PADDLE rispetto alla sinistra alta dello schermo.
Il valore del parametro n e' 0 per leggere una paddlle e 1 per l' altra.

ESEMPIO
Leggere il valore di una paddle in modo diretto.

Azione: Inserire la paddle nella porta giochi del VIC come spiegato nel manualetto consegnato con la periferica.
Girare quindi la manopola della paddle in direzione oraria fino al punto massimo (senza forzare pero'!!).

Digitare:PRINT RPOT(0) (return)

RISULTATO
Viene visualizzato il valore0
Azione: Girare la manopola nella direzione opposta fino in fondo.

Digitare: PRINT RPOT(0) (return)

Risultato: Si visualizza il valore 255.

RPEN

FORMATO: RPEN(n)

FUNZIONE: Legge il valore della LIGHT PEN.

La FUNZIONE RPEN legge la posizione di schermo sulla quale la penna luminosa e' posizionata.
Se il parametro n della FUNZIONE e' n=0 allora sara' letta la posizione relativa alla parte in alto a sinistra (valore x) mentre se n=1 sara' letta la posizione della LIGHT PEN sempre sullo stesso punto dello schermo (valore y).
Il risultato dell' interrogazione, cioe' il valore di questa FUNZIONE sara' un valore da 0 a 255.

ATTENZIONE!!!

Se il valore riportato da questa FUNZIONE eccede il 255 sara' necessario ricalibrare la LIGHT PEN come scritto nelle istruzioni allegate.

ESEMPIO
Leggere il punto sullo schermo sul quale e' posizionata la penna luminosa.

Azione: Collegare la LIGHT PEN al VIC collegandola

alla porta giochi poi scrivere e far girare il seguente programmino:

```
10 X=RPEN(0):Y=RPEN(1):PRINTX,Y:GOTO10
```

posizionando, dopo il RUN, la penna su diversi punti dello schermo.

RISULTATO
In conseguenza al movimento della penna sullo schermo apparira' una serie di coppie di numeri. Il numero a sinistra e' il valore X ed il numero a destra nella coppia e' il valore Y.

## RJOY

FORMATO:RJOY(n)

FUNZIONE:Legge il valore della JOYSTICK

La FUNZIONE RJOY determina il valore che indica il movimento della JOYSTICK.
Come parametro n della FUNZIONE si deve usare un numero qualsiasi compreso fra 0 e 255 che pero' non ha nessun effetto sul valore riportato dalla FUNZIONE.
Il valore generato dalla JOYSTICK quando si muova in una qualsiasi direzione o quando si prema il bottone e' riportato nella seguente tabella:

| DIREZIONE | VALORE |
|---|---|
| Alto | 1 |

| | | | |
|---|---|---|---|
| Basso | | | 2 |
| Destra | | | 8 |
| Sinistra | | | 4 |
| Destra | diagonale | alto | 9 |
| Sinistra | " | " | 5 |
| Destra | " | basso | 10 |
| Sinistra | " | " | 6 |
| Bottone | | | 128 |
| Fermo | | | 0 |

ESEMPIO
Si desideri esaminare tutti i valori possibili della JOYSTICK.

Azione: Inserire la JOYSTICK nella apposita porta giochi del VIC.
Digitare e far girare il seguente programma:

```
10 X=RJOY(0):PRINTX
20 FOR S=1 TO 1000: NEXT
30 GOTO 10
```

Muovere poi la leva in tutte le direzioni e premere il bottone.

RISULTATO
I valori di tutte le posizioni della JOYSTICK e del bottone quando e' premuto, sono visualizzati sullo schermo.

NOTA FINALE

Le funzioni che leggono i valori delle periferiche giochi hanno un uso evidente.Infatti quando si scrivono programmi di giochi e' necessario

conoscere la posizione indicata da ogni giocatore. Questo per quanto riguarda RPOT,RJOY e RPEN.

Il valore delle altre funzioni invece e' evidente nellla prova dei programmi. Infatti se il vostro programma non sta girando correttamente, si potra' per esempio leggere i valori dei colori e dei registri di suono in un punto particolare del programma e vedere se questi valori corrispondono a quelli desiderati.

## NOTE AL HI-RESOLUTION GRAPHIC

Queste note sono state messe a punto grazie alla collaborazione di lettori che ci hanno fatto conoscere i loro problemi nell' uso di questo cartridge.
Ci auguriamo di continuare questa fattiva collaborazione telefonandoci (055/982513) o scrivendoci.

1.0

Con il Cartridge inserito si puo' desiderare di utilizzare i tasti funzione per altri scopi diversi comunque da quelli abilitati dal HI-RESOLUTION.
Sara' allora necessario disabilitare le funzioni gia' assegnate. Ad esempio togliere dal KEY1 il comando GRAPHIC.

Questo puo' essere fatto includendo in un programma i seguenti codici:

```
10 KEY1,CHR$(133)
20 KEY3,CHR$(134)
30 KEY5,CHR$(135)
40 KEY7,CHR$(137)
50 KEY2,CHR$(133)
60 KEY4,CHR$(133)
70 KEY6,CHR$(133)
80 KEY8,CHR$(133)
```

I tasti potranno essere allora letti nel programma con:

```
200 GET A$:IF A$="":THEN200
210 V= ASC(A$):V=1132
220 ON V GOTO500,550,700,750
```

2.0

Il comando CHAR e' utilizzato per visualizzare testi sullo schermo quando ci troviamo in modo grafico (GRAPHIC MODE).
Normalmente questo comando viene utilizzato per visualizzare una costante.
Se si desidera invece visualizzare un variabile N, cio' puo' essere fatto con:

```
CHAR 3,4,STR$(N)
```

Dove 3 stara' ad indicare la riga e 4 la colonna d' inizio.

3.0

E' spesso necessario disconnettere il cartridge del HI-RES.
E' tuttavia noioso e anche pericoloso farlo manualmente.
Ecco come farlo via software:

30 SYS 64850 per disconnettere

30 SYS 41031 per rendere nuovamente operativo HI-RES.

4.0

Usando questo Cartridge non e' sempre possibile utilizzare contemporaneamente sia l' Alta risoluzione che la visualizzazione delle note musicali. E cio' a causa dello scrolling dei disegni.
Le strade per questa combinazione utilissima nei giochi sono due.
La prima e' di ricorrere a dei comandi POKE per la parte musicale. Si tratta tuttavia di un sistema lungo e noioso.
Il secondo sistema e' di stampare i suoni nella parte alta dello schermo utilizzando per esempio un programma come il seguente:

```
90 GRAPHIC2:X=200
95 CIRCLE 2,100,560,35,50
97 PRINT"(home)(ctrl--)V9S203T1GGCD"
100 REGION 2:DRAW2,X,560TO20,560
110      REGION      1      :      Y=Y+20
DRAW1,X-20;560TOX,560:IFX =1024THEN X=200:GOTO97
120 GOTO 100
```

5.0

Uno dei piu' frequenti problemi per gli utenti di un VIC e' quello di far girare un programma scritto con un' espansione da 3K con un' espansione da 8 o piu' K o viceversa.
Le seguenti istruzioni e la tabella sottoriportata aiutano a risolvere il problema.

```
10 POKE642,X:POKE644,Y
20 POKE648,Z:SYS64824
```

In cui i valori X,Y e Z sono da prendersi nella seguente tabella:

| | X | Y | Z |
|---|---|---|---|
| non espanso | 18 | 30 | 30 |
| 3 K | 4 | 30 | 30 |
| 8 K | 18 | 64 | 16 |
| 16K | 18 | 96 | 16 |
| 24K | 18 | 128 | 16 |

6.0

Presto sara' disponibile lo Speech Synthetisers. Tuttavia non sara' possibile farlo funzionare contemporaneamente al HI-RES perche' occupano le stesse locazioni di memoria e gli stessi indirizzi.

## DISEGNO CON LA PENNA LUMINOSA

Per disegnare sul vostro schermo TV con una LIGHT PEN o penna luminosa e' sufficiente oltre a questo poco costoso accessorio il cartridge del SUPER EXPANDER.
Vediamo prima di tutto come il VIC 20 lavora con la penna luminosa.

Il disegno sul vostro schermo TV e' formato da un pennello elettronico che scorre sullo schermo alla velocita' di circa 30 immagini al secondo.
L' integrato 6560 (VIC) presente dentro il computer crea dei segnali che controllano il disegno, in modo tale che sa' DOVE il pennello luminoso si trova in qualsiasi momento.
Una Light pen o penna luminosa contiene un componente sensibile alla luce (fototransistor o fotodiodo) che comunica al 6560 il passaggio nello schermo del pennello elettronico. Le posizioni orizzontali e verticali di questo pennello elettronico vengono immagazzinate in due locazioni di memoria.
Le funzioni RPEN(0) e RPEN(1) del SUPER EXPANDER leggono i valori orizzontali e verticali della Light pen. Gli esperimenti mostrano i limiti e gli intervalli che gli argomenti della funzione RPEN possono assumere quando il SUPER EXPANDER sia fissato in modo grafico. Per chiarezza riportiamo la seguente tabella:

| PUNTO | RPEN(0) | RPEN(1) |
|---|---|---|
| VERTICE ALTO SINISTRA | 34 | 28 |
| VERTICE ALTO DESTRA | 114 | 28 |
| VERTICE BASSO SINISTRA | 34 | 108 |
| VERTICE BASSO DESTRA | 114 | 108 |

La risoluzione dello schermo con una penna luminosa e' di 80 posizioni sia dall' alto in basso sia da sinistra a destra.
Poiche' il SUPER EXPANDER lavora su una scala da 0 a 1023 , il numero letto dalla penna luminosa ha il limite basso sottratto, il che da un intervallo di valori da 0 a 80 . Questo numero e' moltiplicato per 12.8 che allarga l'intervallo da 0 a 1024.
Una penna luminosa di solito usa un interruttore che controlla quando essa in funzione. Questo interruttore e' letto tramite la funzione RJOY(0) che sara' uguale a 1 quando la penna e' in funzione e a 0 quando e' esclusa. Cio' si rende necessario per evitare false letture da parte della penna che risulta sensibile anche a fonti di luce esterne.
Il programma presentato al termine usa il modo grafico multicolore del Vic 20.
I punti sullo schermo possono essere del colore di schermo, del colore del bordo, di un colore ausiliario o del colore del carattere corrispondente a quella zona. Ci puo' essere solo un colore schermo , bordo o ausiliario per volta, ma ci sono 400 diversi colori per ogni casella utilizzabile con il colore carattere.
Cambiando il colore ausiliario cambiera' ogni punto che e' stato disegnato usando quel colore, mentre cambiando solo il colore carattere potremo avere degli effetti di spazi sullo schermo. Alcuni tasti controllano la selezione dei colori e delle speciali funzioni come mostrato nella tavola seguente:

CONTROLLO DI TASTIERA

Selezione dei registri colore:

0 colore di schermo

| | |
|---|---|
| 1 | colore bordò |
| 2 | colore carattere |
| 3 | colore ausiliario |

Cambio dei colori nei registri:

| | |
|---|---|
| CTRL 1 | NERO |
| CTRL 2 | BIANCO |
| CTRL 3 | ROSSO |
| CTRL 4 | CYAN |
| CTRL 5 | PORPORA |
| CTRL 6 | VERDE |
| CTRL 7 | BLEU |
| CTRL 8 | GIALLO |

Cambio dei colori nei registri (solo colori ausiliari o di schermo):

| | |
|---|---|
| Q | ARANCIO |
| W | ARANCIO CHIARO |
| E | ROSA |
| R | CYAN CHIARO |
| T | PORPORA CHIARO |
| Y | VERDE CHIARO |
| U | BLEU CHIARO |
| I | GIALLO CHIARO |

Funzioni speciali:

| | |
|---|---|
| SHIFT/CLR | CLEAR SCREEN |
| F1 | FINE PROGRAMMA<br>CANCELLA SCHERMO |
| P | PAINT |

PROGRAMMA DI DISEGNO CON LIGHT PEN

```
10 GRAPHIC1
20 DIM C(3)
```

```
25 FORL=0TO3
26 C(L)=RCOLOR(L)
27 NEXT
30 XO=RPEN(0)
40 YO=RPEN(1)
60 IFRJOY(0)  1THENR=0:GOTO120
65 X=RPEN(0)
70 Y=RPEN(1)
85 IF(X 34)OR(X 114)OR(Y 28)OR(Y 108)THEN120
86 IFPO=1THENPAINTC,(X-34)*12.8,(Y-28)*12.8:PO=0
88 IFR=0THENPOINTC,(X-34)*12.8,(Y-28)*12.8:R=1
90 DRAWC TO (X-34)*12.8,(Y-28)*12.8
100 XO=X
110 YO=Y
120 GETA$
130 IFA$=""THEN60
140 FORL=1TO23
150 IFA$  MID$("0123 (ctrl1) (ctrl2) (ctrl3)
(ctrl4) (ctrl5) (ctrl6) (ctrl7) (ctrl8) QWERTYUI
(shift/clr) (f1)P",L,1) THENNEXT : GOTO60

160 IFL 5THENC=L-1:GOTO60
170 IFL 13THEN190
175 IFL=21THENSCNCLR:GOTO60
176 IFL=22THENGRAPHIC4:END
177 IFL=23THENPO=1:GOTO60
180 IF(L=0)OR(L=2)THEN60
190 C(C)=L-5
200 COLORC(0),C(1),C(2),C(3)
210 GOTO60
```

## ROUTINES E PROGRAMMI

Sono riportate qui sotto alcune piccole routines d' uso, mentre programmi anche dimostrativi sono riportati al termine del manuale.

```
1   REM***********************
2   REM*                     *
3   REM*  E.V.M  COLOR       *
4   REM*                     *
5   REM*                     *
6   REM***********************
10 REM
50 REM FISSA I COLORI
51 REM
60 GRAPHIC 2
70 COLOR 3,3,0,3
75 REM
80 REM RIEMPE LO SCHERMO CON CARATTERI
85 REM
90 FOR I=0TO19
91 FOR J=0TO19
92 CHAR J,I," "
93 NEXT J
94 NEXT I
95 REM
100 REM FISSA I PARAMETRI
101 REM
105 X=5:Y=5:REM POSIZIONE DI INIZIO
110 S=3:REM MOVIMENTO CARATTERI
120 DIM C(5,5),M(5,5)
125 FOR I=1TO5
130 FOR J=1TO5
135 READ C(J,I)
```

```
140 NEXT J
145 NEXT I
150 DATA 0,0,1,0,0
155 DATA 0,0,1,0,0
160 DATA 1,1,1,1,1
165 DATA 0,0,1,0,0
170 DATA 0,0,1,0,0
190 GOTO 500
195 REM
200 REM INGRESSO MOVIMENTO CARATTERI DA TASTIERA
205 REM
210 A=PEEK(197)
215 XO=X:YO=Y
220 IF A=47 THEN X=X-S:GOTO300
230 IF A=63 THEN Y=Y+S:GOTO300
240 IF A=39 THEN Y=Y-S:GOTO300
250 IF A=55 THEN X=X+S:GOTO300
260 GOTO 210
295 REM
300 REM CONTROLLO CARATTERI
305 REM
310 IF X 5 THEN X=5
320 IF X 165 THEN X=165
330 IF Y 5 THEN Y=5
340 IF Y 165 THEN Y=165
395 REM
400 REM CANCELLA IL CARATTERE
405 REM
410 FOR I=-2TO2
420 FOR J=-2TO2
430 IF M(J+3,I+3)  0 THEN 460
440 POINT 3,(J+XO)*6,(I+YO)*6
445 REM
450 GOTO470
455 REM
460 POINT 4,(J+XO)*6,(I+YO)*6
470 NEXT J
480 NEXT I
495 REM
```

```
500 REM SALVA IL CONTENUTO DI SCHERMO
505 REM
510 FOR I=-2TO2
520 FOR J=-2TO2
530 M(J+3,I+3)=RDOT((J+X)*6,(I+Y*6)
540 NEXT J
550 NEXT I
595 REM
600 REM DISEGNA UN NUOVO CARATTERE
605 REM
610 FOR I=-2TO2
620 FOR J=-2TO2
630 IF C(J+3,I+3)=0 THEN 660
640 POINT 3,(X+J)*6,(Y+I)*6
650 GOTO 670
660 POINT 4,(X+J)*6,(I+Y)*6
670 NEXT J
680 NEXT I
895 REM
900 REM ESEGUI ANCORA
905 REM
910 GOTO 200
```

```
88  REM**********************
89  REM***                ***
90  REM***                ***
100 REM*** TRIDIMENSIONALE***
110 REM***                ***
111 REM***       BY       ***
112 REM***                ***
113 REM***                ***
114 REM***      E.V.M.    ***
115 REM***                ***
116 REM***                ***
117 REM**********************
118 REM
119 REM
120 GRAPHIC 2:COLOR 0,0,5,0
130 A=512:B=A*A:C=512
140 FOR X=OTOA STEP 6
150 S=X*X: P=SQR(B-S):I=-P
160 FOR DO=1 TO 10000
170 R=SQR(S+I*I)/A
180 Q=(R-1)*SIN(24*R)
190 Y=I/3+Q*C
200 IFI=-P THEN M=Y: GOTO230
210 IF Y M THEN M=Y: GOTO240
220 IF Y N THEN 260
230 N=Y
240 Y=C+Y
250 POINT 2,A-X,Y : POINT 2,A+X,Y
260 I=I+4 :IF I P THEN 280
270 NEXT DO
280 NEXT X
290 END
```

## IL LINGUAGGIO MACCHINA

Al centro funzionale e logico di ogni computer e' presente un microprocessore che e' un circuito integrato particolare e costituisce il CERVELLO stesso della macchina.
Il VIC20 non fa eccezione a questa regola ed e' dotato di un collaudato microprocessore il MOS 6502.
Ogni microprocessore ha un suo linguaggio di istruzioni chiamate istruzioni in linguaggio macchina.
Anzi, per essere piu' precisi, il linguaggio macchina o codice macchina e' il solo linguaggio che il VIC20 comprende, cioe' a dire il LINGUAGGIO NATIVO.
La prima domanda che viene da porsi e' quindi come faccia il VIC20 a comprendere un linguaggio come il BASIC che non e' il suo proprio.
Per rispondere a questa domanda dobbiamo per prima cosa vedere cosa accade all' interno del VIC20
A parte il microprocessore e' residente un programma in linguaggio macchina che e' immagazzinato in memoria ROM cosi' che non possa essere cambiato, e cosa molto piu' importante, non si perde allo spegnimento della macchina come invece accade per i programmi scritti da voi ( e che non siano salvati su cassetta o disco).
Questo programma in linguaggio macchina e' chiamato SISTEMA OPERATIVO. Ed il vostro VIC20 sa cosa deve fare all' accensione perche' questo programma gira immediatamente ed automaticamente.
Il Sistema Operativo ha l' INCARICO di organizzare tutta la memoria della vostra macchina per le varie funzioni siano queste il controllo del tasto premuto, la visualizzazione di caratteri, l'

utilizzo delle funzioni e dei comandi.
Infatti quando accendete la macchina il Sistema Operativo esegue una serie di controlli e quindi si dichiara pronto ad operare (READY.)
Un particolare programma dell' intero Sistema Operativo, che dovrebbe essere visto come un insieme di procedure, e' il Basic interprete, cosi' chiamato perche' interpreta ogni comando che diamo alla macchina e se incontra un comando o una funzione che non comprende, cioe' che non riconosce, risponde con un :

?SYNTAX ERROR

A questo punto dovrebbero esservi abbastanza familiari i concetti dei comandi PEEK e POKE cioe' di quei comandi che vanno a leggere e scrivere in determinate locazioni di memoria.
In altre parole avete gia', pur utilizzando comandi e funzioni Basic, scritto e letto DIRETTAMENTE nell' interno del vostro computer sia come memoria che come indirizzi e comandi.
I programmi in linguaggio macchina non sono altro che una serie di istruzioni ad ognuna delle quali puo' o non puo' essere vicino un operando.
Non e' particolarmente difficile utilizzare questo nuovo tipo di linguaggio in modo particolare se si dispone di un programma ASSEMBLER che ne facilita la messa a punto, tuttavia esula dai limiti di questo manuale una trattazione completa di questo linguaggio e del suo utilizzo.
Vi consigliamo di vedere la bibliografia e le notizie utili riportate in questo manuale per approfondire questi concetti.
Poiche' pero' qualcuno potrebbe avere gia' conoscenze di questo linguaggio che e' notevolmente piu' potente del Basic e che

permettendo di utilizzare e conoscere a fondo il computer, riserva delle soddisfazioni che ripagano abbondantemente la fatica di studiarlo, riportiamo in appendice:

LE ISTRUZIONI DELL' ASSEMBLER

LE ROUTINES KERNAL

LE MAPPE DI MEMORIA.

Tuttavia non volendo lasciare l' utente ed il lettore completamente all' oscuro di termini che puo' frequentemente udire o leggere riporteremo alcune brevissime annotazioni tratte dal manuale ASSEMBLER VIC20 -ed. EVM-.
Riprendendo il discorso iniziato in precedenza possiamo affermare che un microelaboratore e' un sistema elettronico in grado di ricevere dall' esterno segnali elettrici, immagazzinarli, elaborarli secondo un certo programma, prendere delle decisioni, naturalmente sempre in base al programma o agli ordini dati ed alle sua intrinseche capacita' operative e quindi emettere dei segnali elettrici utilizzabili all' esterno.

Essenzialmente un micro e' composto da 5 grandi blocchi funzionali:

CPU - Central Processing Unit

cioe' l' unita' centrale di elaborazione, nel nostro caso costituita come detto dal processore 6510.

RAM - Random Access Memory

le memorie di lettura e scrittura.

ROM - Read Only Memory

cioe' memorie a sola lettura

I/O - Input/Output

cioe' tutti i dispositivi di ingresso e uscita e che quindi permettono di colloquire con il mondo esterno sia alla CPU che al computer propriamente detto.

BUS
Insieme di linee sulle quali si muovono i segnali elettrici e che collegano un blocco funzionale all' altro.

Abbiamo detto che all' interno di un microelaboratore circolano dei segnali elettrici che passano da un blocco all' altro attraverso il BUS dei dati.
Questi particolari segnali elettrici si chiamano BIT ed altro non sono che livelli alti o bassi di tensione.
Con la parola BIT che e' la contrazione delle due parole inglesi BINARY DIGIT si intende l' unita' elementare di informazione nell' elettronica ( appunto DIGITALE ) cioe' uno dei possibili stati logici o livelli logici ZERO e/o UNO.

NOTA

Ricordarsi che lo 0 e' usato in una forma di scrittura particolare cioe' con la sbarra in diagonale per distinguerlo dalla lettera O, ma che spesso non e' evidenziato per ragioni tipografiche.

Tornando al nostro discorso dunque il BIT puo' avere due e due soli stati.
Se si considera il classico esempio di una lampadina potremo avere che quest' ultima e' in una delle due condizioni:

Accesa cioe' ON

Spenta cioe' OFF

e di conseguenza con un solo BIT possiamo avere uno dei due stati.
Se avessimo due lampadine ovverosia due BIT potremo avere invece quattro combinazioni diverse:

0 0
1 0
0 1
1 1

Da questo se ne ricava che la formula generale delle combinazioni che possono avere piu' BITS usati contemporaneamente e':

2 elevato alla n

dove n e' il numero di BITS trattati.
Da questo ne consegue che un microprocessore ad 8 BITS, cioe' che puo' trattare fino ad 8 BITS contemporaneamente, puo' ricevere informazioni in 256 combianzioni diverse.

Nel caso del 6510 abbiamo un microprocessore che tratta informazioni a 8 BITS, ma che puo' utilizzare 16 BITS di indirizzi quindi pari a 65536 cioe' 2 alla 16ma.
Abbiamo pero' trovato altri termini per la rappresentazione dei dati:

Nibble cioe' 4 BITS

BYTE cioe' 8 BITS

## I REGISTRI DEL 6502

Sempre senza la pretesa di approfondire vediamo quali sono i registri principali del nostro microprocessore.

## L' ACCUMULATORE

Come dice il nome stesso e' un' area di immagazzinamento.
E' il piu' importante registro del microprocessore ed e' controllabile tramite una nutrita serie di comandi in codice macchina.
Questi comandi o istruzioni consentono di copiare il contenuto di una locazione di memoria nell' Accumulatore, di eseguire l' operazione inversa, di modificarne i contenuti e di accdervi direttamete senza effetto sulle altre locazioni di memoria.

## I REGISTRI INDICE X , Y

Altre due locazioni di memoria che potremo definirre complementari o di aiuto alle funzioni dell' Accumulatore e che comunque consentono una notevole agilita' operativa.
Le operazioni consentite su questi registri sono simili a quelle relative all' Accumulatore ma altre devono essere svolte SOLO da uno o tutti e due i registri.

## PROGRAM COUNTER

Un programma in codice macchina consiste esenzialmente in una serie di istruzioni come:

Prendi il contenuto della locazione di memoria X

Sommalo al contenuto della locazione di memoria Y

Metti il risultato dnella locazione di memoria Z.

Questo, come del resto altri programmi piu' complessi, puo' essere eseguito dalla CPU.
E' evidente pero' che la CPU cioe' il processore non puo' iniziare ad eseguire istruzioni partendo da un punto qualsiasi della memoria e seguendo un ordine casuale, ma deve iniziare partendo da un punto ben preciso che rappresenta l' inizio del programma.
Notare che questo avviene anche nell'. esecuzione di un programma scritto in Basic.
Nella CPU si trova un registro chiamato PROGRAM COUNTER o contatore di programma che contiene la

locazione di memoria dalla quale verra' prelevata la prossima istruzione del programma che la CPU stessa deve eseguire.
In linguaggio tecnico questo registro viene chiamato anche POINTER cioe' puntatore perche' punta, cioe' guarda alla prossima istruzione da eseguire e viene abbreviato cone le lettere PC.
Dopo aver eseguito un' istruzione del programma il PC si posiziona automaticamente all' indirizzo dell' istruzione da eseguire immediatamente dopo e quindi non si richiede nessun intervento da parte dell' utente il quale tuttavia potra' eventualmente modificarne il contenuto per far SALTARE il programma non all' istruzione immediatamente seguente ma da altra parte.
Come si vede nelle tavole riportate in fondo, le istruzioni del 6502 non sono tutte di una sola locazione di memoria, ma possono interessare da 1 a 3 locazioni. In questo caso il PC si incrementara' di conseguenza.
Abbiamo insistito in modo particolare su questo regsitro di 16 BITS per le analogie che presenta con i relativi registri di pagina ZERO del Basic che come abbiamo detto hanno funzioni similari.

## LO STATUS REGISTER

Anche questo e' un registro importantissimo nell' architettura della CPU.
Ognuno degli otto BITS che lo compongono e che in questo caso sono denominati FLAGS ha una funzione particolare ed e' programmabile con particolari comandi.

Come gia' ricordato per approfondire l' argomento e' necessario un apposito manuale come quelli che trovate nella bibliografia di questo volume.
Per questo motivo ricordiamo che l' uso del VICMON potra' essere utile solo a chi possieda una certa conoscenza dell' Assembler ecc.

## IL LINGUAGGIO MACCHINA : APPROFONDIMENTO

In questo capitolo desideriamo approfondire alcuni concetti essenziali relativamente al Monitor implementato, essenzialmente:

- I MODI DI INDIRIZZAMENTO
- IL REGISTRO DI STATUS
- IL PROGRAM COUNTER
- LO STACK REGISTER

### I MODI DI INDIRIZZAMENTO

Ciascuna istruzione di un programma in linguaggio macchina contiene le informazioni sulla posizione dei dati sui quali l' informazione stessa dovra' poi operare.
La stessa istruzione puo' essere messa sotto diverse forme in dipendenza dal punto in cui queste istruzioni sono localizzate. Ognuna di queste forme e' riferita in particolare all' indirizzamento utilizzato.
Esistono 13 diversi modi di indirizzamneto e molte istruzioni possono essere date in piu' di un modo.
Ad esempio l' istruzione LDA puo' essere indirizzata in 8 diversi modi.
Questi vari modi possono essere divisi in 7 modi base e sei modi composti che sono la combinazione di uno dei modi base insieme ad un modo di indirizzamento cosi' detto "INDEXATO".

La seguente tabella mostra i vari modi.
I modi base sono:

- IMPLICITO
- ACCUMULATORE
- IMMEDIATO
- ASSOLUTO
- IN PAGINA ZERO
- RELATIVO
- INDIRETTO

Questi altri modi sono i modi combinati di indirizzamento:

-* ASSOLUTO X INDICIZZATO

-* ASSOLUTO Y INDICIZZATO

-* PAGINA ZERO X INDICIZZATO

-* PAGINA ZERO Y INDICIZZATO

-* INDIRETTO INDICIZZATO

Il piu' semplice modo di indirizzamento e' il MODO IMPLICITO, che come dice la parola e' insito nello stesso comando.
E' usato infatti per istruzioni di un singolo Byte

che operino direttamente verso l' inerno dei registri del processore.

In una istruzione come CLC (clear Carry) non e' necessario accedere a nessun dato ragion per cui non e' richiesto nessun indirizzo.

Esempio:

CLC Setta il Carry a O
INX Incrementa di 1 il registro X
RTS Ritorno da Subroutine

Il modo di indirizzo Accumulatore e' usato in istruzioni che consentono operazioni logiche sui dati nell' Accumulatore.
Questo modo e' una versione particolare dell' Indirizzo Implicito e tutte le istruzioni sono di un singolo Byte.

Il modo di indirizzamento IMMEDIATO e' usato ogni volta che il programmatore desidera eseguire una operazione usando una costante. Per mettere un valore, ad esempio 25, nell'Accumulatore bisogna utilizzare l' istruzione LDA in modo Immediato.

Esempio:

LDA £$00 Carica 00 nell' Accumulatore
AND £$FF Esegue un AND logico fra il valore FF ed il contenuto dell' Accumulatore.

In questa forma di indirizzamento il dato e' immagazzinato nel Byte immediatamente seguente l' OPCODE ( o codice operativo).

Ne' il modo di indirizzo immediato ne' il modo di

indirizzamento IMPLICITO usano un indirizzo di memoria per immaggazzinare i dati e quindi sono di uso molto modesto per operazioni con delle variabili.

Per indirizzare una qualsiasi locazione di memoria saranno necessari due Bytes di indirizzo immagazzinati nella parte operando dell' istruzione.
Questi indirizzi indicano una locazione di memoria dove la variabile sopra alla quale deve esere eseguita l' operazione e' collocata in quel preciso momento o dove deve essere immagazzinata.
Questa forma di indirizzamento e' nota come "forma di indirizzamento assoluto".

Esempio:

LDA $1000 Carica nell' Accumulatore il contenuto della locazione $1000

STA $0401 Carica nella locazione di indirizzo $0401 il contenuto dell' Accumulatore.
Una forma particolare di indirizzamento assoluto puo' essere usata quando la locazione di memoria alla quale si deve accedere e' situata nella Pagina Zero della memoria.

Questo e' il solo caso in cui il concetto di pagina abbia una qualche importanza nel micro 6502.
Infatti la pagina Zero occupa le prime 256 locazioni di memoria.

Questo sistema e ' chiamato :

INDIRIZZAMENTO IN PAGINA ZERO

ed usa un solo Byte di indirizzo per puntare ad una locazione di dati in pagina Zero. Questo e' possibile perche' in questo caso esiste appunto l' implicita premessa che ci si riferisce alla pagina zero.
Una istruzione di indirizzo in pagina zero quindi di due Byte e' molto piu' veloce che la stessa istruzione di indirizzo, indirizzo ovviamente in modo Assoluto, di tre Bytes, ed e' buona pratica di immagazzinare tutte le variabili in pagina Zero, cosa che ha fatto chi ha scritto il Sistema Operativo del VIC.

Quando sta girando un programma in linguaggio macchina sul VIC solo i primi 143 bytes della pagina zero dovrebbero essere usati per immagazzinare i data ,perche' immagazzinando dati nelle locazioni superiori della pagina zero si puo' causare un blocco del sistema.

Una forma speciale di indirizzamento e' usata eclusivamente per eseguire operazioni di deviazione e di salto, ed e' conosciuta come "INDIRIZZAMENTO RELATIVO".
In questo modo di indirizzamento la istruzione e' seguita da un operando di un singolo bytes.

Cio' non specifica un indirizzo come nel modo di indirizzamento in pagina zero, ma uno spostamento di indirizzo dove l' istruzione di deviazione e' immagazzinata. Poiche' lo spostamento puo' essere positivo o negativo l' ottavo bit e' usato per spiegare la direzione di salto cio' consente che questo sia di 127 bytes in avanti o di 128 bytes in dietro.

In alcuni programmi puo' essere necessario di avere un indirizzo calcolato invece di un

indirizzo fisso come invece avverrebbe nell' indirizzamento assoluto.
Questo si puo' fare usando un modo di indirizzamento indiretto.In questo modo le istruzioni hanno un byte di indirizzo poiche' gli indirizzi dei dati non sono immagazzinati direttamente nell' operando dell' istruzione ma direttemante in pagina zero. Tutti gli accessi indiretti sono indicizzati fuorche' per le istruzioni di JMP.

Esempio:

JMP ($3000) Eseguira' un salto prendendo come indirizzo il valore contenuto nella locazione indicata nel parametro.

Se per esempio dando il comando :

M 3000

avessimo come risultato:

06 23 12 45 21

allora:

JMP ($3000) legge 06 23 e quindi saltera' alla locazione:

$2306

L' indirizzamento indicizzato usa i contenuti di uno o di due registri indice come una devizione agli indirizzi immagazzinati nell' operando della parte dell' istruzione. L' indirizzo immagazzinato nell' operando puo' essere sia un indirizzo

assoluto di due bytes oppure un indirizzo in pagina zero di un byte.
Cio' da' un totale di quattro modi di indirizzamento indicizzato, due per ogni registro indice. L' uso principale dell' indirizzamento indicizzato e' nella possibilita' di accedere a successive locazioni di memoria usate per l' immagazzinamento di tavole o blocchi di data.

Nel modo di indirizzamento INDIRETTO INDICIZZATO, il registro indice X e' aggiunto all' indirizzo dell' operando in pagina zero che punta alle locazioni dove i sedici bit di indirizzo DATA sono immagazzinati.

Nel modo di indirizzamento INDICIZZATO INDIRETTO per prima cosa e' trovato il puntatore di indirizzo a 16 bit situato in pagina zero, successivamente deviato dal contenuto del registro indice Y che da il VERO indirizzo dei dati.
La locazione del puntatore e' fissa , mentre invece in questo modo diviene una variabile essendo deviata dal contenuto del registro indice X.

Quest' ultimo modo di indirizzamento combina i vantaggi di indirizzare qualsiasi punto della memoria con i vantaggi di deviazione forniti dai contenuti dei registri indice.

## IL REGISTRO DI STATUS E L' USO DEI FLAGS.

Il registro di processo occupa una posizione molto importante nell' architettura di un sistema basato sul microprocessore 6502.
Questo e' un registro programmabile di otto bit, tuttavia a differenza degli altri registri la sua funzione e' posta fra la sezione di controllo e la sezione di registro del processore.
E' il solo registro che attui controlli logici. Sette dei suoi otto bits o FLAGS sono usati ed ognuno ha una specifica funzione.
I flags di questo registro si dividono in tre categorie.
Alcuni controllabili solo tramite programma, altri controllabili sia tramite programma sia tramite processore e gli ultimi che vedremo sottoposti SOLO al controllo del microprocessore.
Solo uno di questi bit o flag rientra nella prima categoria ed e' esattamente il D flag o DECIMAL MODE REGISTER che occupa il bit n. 3 del REGISTRO DI STATO.
Questo flag controlla che il processore esegua le operazioni aritmetiche in binario o in decimale.
Come abbiamo visto usando l' istruzione SED, tutte le operazioni vengono eseguite in modo decimale fino a quando il Flag D non sia settato tramite un istruzione CLD o Clear Decimal Mode.

I flags che appartengono alla seconda categoria, cioe' quelli controllabili sia dal programmatore che dal microprocessore sono i seguenti:

IL CARRY

L' OVERFLOW

L' INTERRUPT DISABLE.

Il Carry o C-Flag e' localizato nel bit zero del Registro di Stato ed e' modificato sia dal risultato di determinate operazioni aritmetiche sia dal programmatore.
Il Carry e' anche usato come nono bit durante le operazioni aritmetiche o per le istruzioni di SHIFT o di ROTATE.
L' istruzione usata per settare il flag di Carry e' l' istruzione SEC, mentre per azzerare questo bit si adopera il comando CLC.

L' Overflow o Flag V occupa il Bit n. 6 del registro di stato ed e' usato durante operazioni aritmetiche con il segno per indicare che il risultato era di valore piu' grande di quanto poteva essere contenuto nel settimo bit del Byte specificato.
L' Overflow ha lo stesso significato del Flag C o bit di Carry, ma indica anche che una routine di correzione segno deve essere usata se questo bit e' in posizione di ON fino a quando l'Overflow stesso non avra' cancellato il segno nel settimo bit.
Solo il programmatore puo' riportare a zero il Flag V usando l'istruzione CLV.

Il bit di INTERRUPT DISABLE o I Flag controlla le operazioni del microprocessore durante le richieste di interrupt ed si trova nel bit n. 2 del Registro di stato.
Come vedremo in seguito gli interrupts hanno un ruolo molto importante nel funzionamento del VIC e tutte le volte che c'e' un Interrupt il processore setta l' I Flag.
L' I Flag puo' essere settato dal programmatore

con l' istruzione SEI per prevenire una interruzione sul lavoro del microprocessore come durante, ad esempio, una istruzione di LOOP.
Alla fine di un programma o di una routine come quella che abbiamo detto prima la linea di Interrupt puo' essere riportata alla sua normale funzione riportando a zero il Flag I CLI.

Gli ultimi tre registri sono:

ZERO

NEGATIVE

BREAK

e sono controllati esclusivamente dal microprocessore.

Il Bit Zero o Z flag e' settato dal processore ogni qual volta il risultato di una operazione e' zero come quando per esempio si sottrae l' uno da l' altro due numeri dello stesso valore.
Il Bit Negative o N Flag e' posto usuale al settimo bit dal processore in base al risultato di una operazione; uno degli usi piu' importanti di questo Flag e' durante operazioni aritmetiche in binario con segno.
Infatti se e' settato l'N flag il risultato sara' allora un numero negativo.
Il Bit di Break o B Flag e' settato dal processore durante una sequenza di Interrupt.
Il Flag Z occupa il bit n 1, il flag N il bit 7 ed il Flag B il bit n 4 del Registro di Stato.

I sette Bit o Flags dello status register hanno ognuno un significato per il programmatore in un

particolare punto del programma.
Sebbene il Carry e l' Overflow siano presenti nelle operazioni aritmetiche il maggior uso dei Flags e' in combinazione con le operazioni di salto condizionato.
Cio' da al programmatore la possibilita' di incorporare gruppi di istruzioni decisionali entro un programma.
Per testare uno di questi bit e' necessario eseguire un 'azione come quelle che riportiamo.
Un salto condizionato ha la stessa funzione del IF.....THEN.....GOTO del Basic, per cui ci sono un gruppo di queste istruzioni che eseguono diverse funzioni e testano i differenti Flags.

## LE DEVIAZIONI, I SALTI ED IL PROGRAM COUNTER

Per comprendere l' uso delle istruzioni di deviazione e di salto e' necessario prima spiegare il concetto di sequenza nel programma e il suo controllo tramite un contatore dei passi di programma o PROGRAM COUNTER.
In pratica si tratta di definire il concetto di subroutine.
Il contatore di programma o PROGRAM COUNTER consiste di due registri di otto bit.
Come gli altri registri, anche questo comunica con il processore attraverso il Data Bus ma l' uscita e' anche connessa a 16 linee del bus indirizzi.
Uno dei registri del PROGRAM COUNTER e' connesso alle linee di indirizzo ( le otto piu' alte ) ed e' chiamato PROGRAM COUNTERH, mentre l' altro e' connesso alle otto linee piu' basse e si chiama PROGRAM COUNTERL.
Sebbene si tratti di due registri da otto bit ciascuno il funzionamento del PROGRAM COUNTER e' perfettamente identico ad un registro da sedici bit.
E' il PROGRAM COUNTER che controllera' l' indirizzamento della memoria o i puntatori dell' indirizzo dato perche' contiene l' indirizzo della successiva locazione di memoria che deve essere esaminata dal processore.

All' inizio del programma il PROGRAM COUNTER conterra' l' indirizzo della prima istruzione. Questa e' una delle funzioni dell' OPERATING SISTEM RESET SOFTWARE ed e' anche eseguita dai comandi SYS e USR quando si passa da un programma BASIC ad un programma in Linguaggio Macchina.

L' istruzione caricata dalla memoria e'

immagazzinata in un registro di istruzioni per essere decodificata e quindi resa COMPRENSIBILE, dall' unita' di controllo.
Questo processo richiede un ciclo di CLOCK (per vedere che cosa sia il Clock e come operi vedere il capitolo relativo nella GUIDA AL VIC 20) , durante questo periodo il Program Counter e' incrementato di UNO e punta quindi alla successiva locazione di memoria.
Di norma il processore richiede piu' di un Byte per interpretare un' istruzione.
Il primo Byte contiene l' operazione basa ed e' conosciuto come OPERATION CODE o OP CODE mentre i successivi Bytes conosciuti come OPERANDS possono contenere sia un Byte di dati o l' indirizzo al quale sara' necessario accedere per questi dati.

Un' istruzione puo' richiedere fino a tre locazioni di memoria sequenziali per il PROGRAM COUNTER che prima di tutto punta all' OPERATION CODE che e' andato a prelevare dalla memoria e lo immagazzina nel registro di istruzione.
Il PROGRAM COUNTER e' quindi incrementato e punta alla successiva locazione di memoria, i contenuti della quale sono riportati e immagazzinati nella ALU (Aritmetic Logic Unit).

Dopo aver completato l' operazione che usualmente richiede circa quattro cicli di Clock, il processore incrementa il PROGRAM COUNTER per puntare alla successiva istruzione e il processo si ripete.
In questo modo il PROGRAM COUNTER continuera' ad avanzare fino alla locazione massima di memoria riportando istruzioni ed indirizzi, o fino ad uno stop.

## LO STACK REGISTER ED IL SUO USO

Lo Stack Register e' principalmente usato per la manovra degli Interrupts e delle Subroutines.
E' un registro di otto bits la cui funzione e' identica a quella del PROGRAM COUNTER.
E' usata per puntare ad un indirizzo in pagina UNO della memoria (locazioni decimali da 256 a 511) conosciuta come " AREA DI STACK ".
Lo Stack e' settato quindi dalle locazioni di memoria che iniziano dal decimale 511 indietro fino ad un massimo di 255 Bytes. E' un registro che viene utilizzato con il metodo LIFO cioe' Last In First Out.
Cio' consente che l' ultimo bit immagazzinato nello Stack sia il primo al quale si ha accesso.
Tutte le volte che un dato e' inserito nel' area di Stack lo Stack Pointer e' decrementato di uno, mentre invece quando ne esce lo Stack Pointer e' incrementato di uno.
Il metodo di indirizzamento dello Stack e' indipendente dal programma ed e' basato unicamente su una serie di eventi cronologici.
Lo Stack in effetti e' usato come una zona di immagazzinamento temporaneo.
Tutte le volte infatti che e' chiamata una Subroutine in un programma in Linguaggio Macchina l' attuale contenuto del Program Counter deve essere salvato.
Al ritorno dalla subroutine il programma deve ripartire dalla corretta locazione. Nello stesso modo tutte le volte che e' interrotto il funzionamento del microprocessore, l' indirizzo corrente e' salvato nel PROGRAM COUNTER prima che il microprocessore faccia eseguire la routine di Interrupt.
Una Subroutine puo' richiamare altre Subroutines

che richiedono quindi l' immagazzinamento di piu' indirizzi nello Stack.
L' ultimo indirizzo di rientro immagazzinato sara' il primo indirizzo richiamato nel Program counter alla fine della Subroutine, di qui la necessita' di struttura LIFO dello Stack Pointer.
Il sistema di chiamare delle Subroutines tramite altre Subroutines e' chiamato " SUBROUTINES NESTING " ed e' molto comune nei programmi in Linguaggio Macchina.
L' organizzazione dello Stack nel 6502 limita l' utente ad un massimo di 127 livelli di Nesting che e' in effetti una cifra maggiore di quanto normalmente si possa usare.
Le Subroutines in Basic usano lo Stack per immagazzinare il ritorno di indirizzi ed i contenuti dei registri.
Una subroutine e' richiamata tramite una istruzione JSR o Jump To Subroutine cioe' Salta alla Subroutine.

Questo comando spinge il contenuto corrente contenu del PC entro lo Stack.
Una locazione di memoria immagazzinata come OPERAND FIELD viene quindi immgazzinata entro il PC. Cio' induce il Micro a saltare ad una nuova sub del programma ed a iniziare l' esecuzione da una locazione immagazzinata nel PC.
Il rientro da una subroutine al programma principale e' eseguito attraverso una istruzione RTS o Return From Subroutine cioe' ritorno da Subroutine.
L' esecuzione di questo comando carica l' indirizzo di ritorno dallo Stack Pointer entro il Program Counter ed incrementa il PC per puntare all' istruzione seguente il comando JSR.
Anche lo S P e' incrementato per puntare al successivo indirizzo di subroutine se ne esiste uno.

Lo Stack puo' essere usato dal programmatore come una locazione di immagazzinamento temporaneo per i dati passati alla subroutine.
Al programmatore infatti necessitano un gruppo di istruzioni che gli consentano di mettere questi dati entro lo S e poi di rileggerli.
Il contenuto corrente dell' Accumulatore puo' essere trasferito alla successiva locazione di memoria nello Stack Register attraverso un comando di PHA o Push Accumulator On To Stack.

I dati possono essere letti dalla corrente locazione puntata attraverso lo Stack Pointer entro l' Accumulatore attraverso il comando PLA o Pull Accumulator From Stack.

Tutte queste istruzioni provvedono che lo Stack Pointer sia automaticamente incrementato o decrementato di uno.

Un esempio di immagazzinamento dati nello Stack e' di poter salvare il contenuto dello Status Processor e dei registri indice quando viene chiamata una subroutine.
I contenuti dello status register possono essere inseriti entro lo Stack tramite i comandi PHP o Push Processor Status Onto Stack.
Succesivamente possono essere ritrasferiti dallo Stack allo Satus Register attraverso il comando PLP o Pull Processor Status From Stack.
Per salvare il contenuto dei registri indice bisogna trasferire all' accumulatore il contenuto di questi registri e dopo inserirli nello Stack.
Quando si sta scrivendo una routine in Linguaggio Macchina per il VIC che sara' successivamente richiamata da un programma Basic e' molto importante di salvare prima di tutto il contenuti del Processor Accumulator e dei registri indice nello Stack.

I contenuti di questi registri infatti saranno succesivamente rimessi a punto prima di tornare al Basic in quanto se cio' non avvenisse potrebbe causare danni al programma.
Normalmente lo S P punta ad una locazione in pagina uno e questa locazione sara' automaticamente incrementata o decrementata da ordini del processore.
In alcuni casi il programmatore ha necessita' di cambiare il contenuto dello Stack Pointer.
Allora lo Stack Pointer e' caricato tramite il trasferimento dei contenuti del Registro indice X (operazione che avviene tramite l' istruzione TXS o Transfer Index X To Stack Pointer).
Questa istruzione e' usata all' inizio di un Programma per inizializzare lo Stack Pointer ed e' eseguita automaticamente sul VIC come parte della routine di POWER UP.
La preinizializzazione dello Stack sul VIC puo' causare dei problemi allora sara' necessario rileggere il corrente contenuto dello Stack Pointer e cercarlo nuovamente sul Registro indice X con l' istruzione TSX o Trasfer Stack Pointer to Index X.

## I REGISTRI INDICE

L' avere un indirizzo fisso in un campo operativo di una istruzione pone dei problemi quando si debba accedere ad un blocco di dati sequenziali come potrebbe essere una tavola o un buffer in posizione di Input.
Uno potrebbe essere di usare una stringa per caricare le istruzioni in una certa forma:

Carica i dati dall' indirizzo 1

Esegui l' operazione

Carica i dati dall' indirizzo 2

Esegui l' operazione

e cosi' via.

Questo occupa una larga parte di spazio della memoria e oltre tutto a discapito della efficienza e della rapidita' di esecuzione del programma.
Un approccio piu' sofisticato e' nell' uso di contatori il contenuto dei quali sia automaticamente aggiunto all'indirizzo del campo operando dell' istruzione.
Due contatori chiamati anche registri indice sono presenti nel 6502 ambedue di otto bit:

REGISTRO INDICE X

REGISTRO INDICE Y

Questi sono usati tramite istruzioni in uno dei modi di indirizzo indicizzati.
Il piu'semplice e' il modo di indirizzamento indicizzato assoluto nel quale il contenuto di un registro indice e' aggiunto all' indirizzo del campo operando dell' istruzione, per cui si ottiene un nuovo indirizzo al quale si potra' accedere per i DATA.

Il fatto che un registri indice sia di solo otto bit limita la dimensione massima del blocco data accessibile, usando l' indirizzo indicizzato, a 256.
In pratica pero' la maggior parte delle tavole e' piu' corta per cui non e' una significativa limitazione.

Se e' necessario manipolare tavole piu' lunghe allora le tecniche di programmazione come l' indirizzo indiretto indicizzato serviranno a superare questa limitazione.
I registri indice sono controllati e manovrati da un gruppo di istruzioni particolari.
Un numero puo' essere caricato dentro o immagazzinato o richiamato dal registro indice tramite le istruzioni LDX, LDY, STX, STY.
Nello stesso modo il contenuto dei registri indice puo' essere confrontato con valori in memoria per vedere se e' necessario un salto condizionato ad una determinata posizione usando le istruzioni CPX e CPY.
Il contenuto di un qualsiasi registro indice e' modificato per puntare alla succsiva locazione incrementando lo stesso registro di uno o decrementandolo.
Per incrementare questi registri useremo le istruzioni INX o INY e per decrementarli DEX o DEY.

Le rimanenti istruzioni relative ai registri indice consentono il trasferimento dei contenuti dell' Acccumulatore in uno dei registri indice e Viceversa.
Si tratta ddelle istruzioni TAX e TAY che trasferiscono il contenuto dell' Accumulatore dentro i registri X e Y oppure delle istruzioni TXA e TYA che trasferiscono i contenuti dei registri indice nell' Accumulatore.
In alcuni programmi si rende necessario avere uno indirizzo calcolato piuttosto che avere un indirizzo di base con una deviazione, come nel modo di Indirizzo Indicizzato Assoluto.
Cio' e' eseguito usando l' indirizzamento indiretto in cui le istruzioni hanno appena un Byte di Campo indirizzi che puntera' all' indirizzo effettivo come due Bytes in pagina Zero.
L' indirizzo dati non sara' allora immagazzinato direttamente nel campo operandi dell' istruzione ma indirettamente in pagina zero e tutti gli indirizzi di accesso saranno indicizzati eccetto che per gli indirizzi di Jump o di salto.
Ci sono due modi disponibili di indirizzamento indiretto:

INDIRIZZAMENTO INDIRETTO INDICIZZATO

INDIRIZZAMENTO INDICIZZATO INDIRETTO

Il primo metodo indica che il registro X debba essere aggiunto all' indirizzo dell' operando in pagina zero.
Questo punta alla locazione dove l' indirizzo dei sedici dati e' immagazzinato.Uno dei piu' larghi usi di questo modo di indirizzamento e' di ritrovare dei dati da una tabella o da una lista di indirizzi.

Il secondo metodo di indirizzamento funziona in questo modo.
Per primo vengono letti i sedici bit di indirizzo in pagina zero successivamente l' indirizzo e' deviato dal contenuto del registro indice Y per dare il vero indirizzo dei dati.
La locazione del puntatore e' fissa tuttavia in questo modo diventa variabile a causa della deviazione effettuata tramite il contenuto del registro X.
Questo modo di indirizzamento combina il vantaggi di un indirizzo che puo' puntare da qualsiasi parte della memoria con la capacita' di deviazione fornita dai registri indice.

E' un metodo particolarmente potente per accedere all' ennesimo elemento di una tavola il cui indirizzo di partenza sia immagazzinato in pagina Zero.

# APPENDICI

# ISTRUZIONI IN ORDINE ALFABETICO DEL 6502

| | |
|---|---|
| **ADC** | Somma con riporto |
| **AND** | AND Logico |
| **ASL** | Spostamento Aritmetico a Sinistra |
| **BCC** | Opera diramazione se carry è zero |
| **BCS** | Opera diramazione se carry è uno |
| **BEQ** | Opera diramazione se risultato = 0 |
| **BIT** | Verifica di bit |
| **BMI** | Opera diramazione se negativo |
| **BNE** | Opera diramazione se diverso da 0 |
| **BPL** | Opera diramazione se positivo |
| **BRK** | Break |
| **BVC** | Opera diramazione se overflow è 0 |
| **BVS** | Opera diramazione se overflow è 1 |
| **CLC** | Azzera carry |
| **CLD** | Azzera il flag decimale |
| **CLI** | Azzera la disabilitazione interrupt |
| **CLV** | Azzera overflow |
| **CMP** | Confronta con l'accumulatore |
| **CPX** | Confronta con X |
| **CPY** | Confronta con Y |
| **DEC** | Decrementa la memoria |
| **DEX** | Decrementa X |
| **DEY** | Decrementa Y |
| **EOR** | OR Esclusivo |
| **INC** | Incrementa X |
| **INY** | Incrementa Y |
| **JMP** | Salta |
| **JSR** | Salta alla subroutine |
| **LDA** | Carica l'accumulatore |
| **LDX** | Carica X |
| **LDY** | Carica Y |
| **LSR** | Spostamento logico a destra |
| **NOT** | Non opera |
| **ORA** | OR Logico |
| **PHA** | Introduce A |
| **PHP** | Introduce lo stato P |
| **PLA** | Estrae A |
| **PLP** | Estrae lo stato P |
| **ROL** | Rotazione a sinistra |
| **ROR** | Rotaziona a destra |
| **RTI** | Ritorno da Interrupt |
| **RTS** | Ritorno da subroutine |
| **SBC** | Sottrae con riporto |
| **SEC** | Pone carry ad 1 |
| **SED** | Pone decimale ad 1 |
| **SEI** | Pone disabilitazione interrupt ad 1 |
| **STA** | Immagazzina l'accumulatore |
| **STX** | Immagazzina X |
| **STY** | Immagazzina Y |
| **TAX** | Trasferisce A in X |
| **TAY** | Trasferisce A in Y |
| **TSX** | Trasferisce SP in X |
| **TXA** | Trasferisce X in A |
| **TXS** | Trasferisce X in SP |
| **TYA** | Trasferisce Y in A |

# TAVOLA DI CONVERSIONE ESADECIMALE/DECIMALE

| 5 | | 4 | | 3 | | 2 | | 1 | | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC |
| 0 | 0 | 0 | | 0 | | 0 | | 0 | 0 | 0 | 0 |
| 1 | 1,048,576 | 1 | 65,536 | 1 | 4,096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 2,097,152 | 2 | 131,072 | 2 | 8,192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 3,145,728 | 3 | 196,608 | 3 | 12,288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 4,194,304 | 4 | 262,144 | 4 | 16,384 | 4 | 1,024 | 4 | 64 | 4 | 4 |
| 5 | 5,242,880 | 5 | 327,680 | 5 | 20,480 | 5 | 1,280 | 5 | 80 | 5 | 5 |
| 6 | 6,291,456 | 6 | 393,216 | 6 | 24,576 | 6 | 1,536 | 6 | 96 | 6 | 6 |
| 7 | 7,340,032 | 7 | 458,752 | 7 | 28,672 | 7 | 1,792 | 7 | 112 | 7 | 7 |
| 8 | 8,388,608 | 8 | 524,288 | 8 | 32,768 | 8 | 2,048 | 8 | 128 | 8 | 8 |
| 9 | 9,437,184 | 9 | 589,824 | 9 | 36,864 | 9 | 2,304 | 9 | 144 | 9 | 9 |
| A | 10,485,760 | A | 655,360 | A | 40,960 | A | 2,560 | A | 160 | A | 10 |
| B | 11,534,336 | B | 720,896 | B | 45,056 | B | 2,816 | B | 176 | B | 11 |
| C | 12,582,912 | C | 786,432 | C | 49,152 | C | 3,072 | C | 192 | C | 12 |
| D | 13,631,488 | D | 851,968 | D | 53,248 | D | 3,328 | D | 208 | D | 13 |
| E | 14,680,064 | E | 917,504 | E | 57,344 | E | 3,584 | E | 224 | E | 14 |
| F | 15,728,640 | F | 983,040 | F | 61,440 | E | 3,840 | F | 240 | F | 15 |

Appendice 3

| HEX | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 2 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 3 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 4 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 5 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 6 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 7 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 8 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| 9 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| A | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| B | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| C | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| D | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| E | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| F | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

TAVOLA DI CONVERSIONE ESADECIMALE/DECIMALE

**Appendice 4**

```
1000 REM-----------------------------------
1010 REM PROGRAMMA PER LA VISUALIZZAZIONE
1020 REM DI UN FUNZIONE MATEMATICA
1030 REM INSERITA NELLA LINEA 1270
1040 REM ----------------------------------
1050 PRINT"(CLR/HOME)"
1060 PRINT"INSERISCI IL LIMITE
1070 PRINT"INFERIORE  ";:INPUTX1
1080 PRINT"QUELLO SUPERIORE";:INPUTX2
1090 X=X1:GOSUB1270
1095 PRINT"VAL. DI Y ";INT(Y)" PER X="X1
1100 X=X2:GOSUB1270
1105 PRINT"VAL. DI Y ";INT(Y)" PER X="X2
1110 PRINT"L' INCREMENTO   ";:INPUTS
1120 PRINT"FATTORE INCR. X ";:INPUTFX
1130 PRINT"FATTORE INCR. Y ";:INPUTFY
1140 GRAPHIC2:REM ENTRO IN MODO GRAFICO
1145 REM ********************
1148 REM TRACCIO L' ASSE X
1150 DRAW2,0,500TO1023,500
1155 REM ********************
1157 REM
1158 REM ********************
1159 REM TRACCIO L'ASSE Y
1160 DRAW2,500,0TO500,1023
1165 REM ********************
1170 FORW=X1TOX2STEPS
1180 X=W:GOSUB1270
1190 Y=Y*FY:X=X*FX
1200 GOSUB1290
1210 POINT2,XX,YY
1220 NEXT
1230 GETX$:IFX$=""THEN1230
1240 GRAPHIC0:GOTO1040
1270 Y=SIN(X)*2
1280 RETURN
1290 REM ---- SPOSTAMENTO ASSI ----
1300 XX=X+500
1310 YY=(Y*-1)+500
1320 IFXX<0THENXX=0
1330 IFYY<0THENYY=0
1340 RETURN
```

```
10 REM ---- CALEIDOSCOPIO ----
30 GRAPHIC1
40 DIMA(1),D(1),C(3),X(5),Y(5),X1(5),Y1(5)
50 DRAW1,255,0TO0,511TO255,1023
60 DRAW1,767,0TO1023,511TO767,1023
70 PAINT1,0,0
80 PAINT1,0,1023
90 PAINT1,1023,0
100 PAINT1,1023,1023
110 X=RND(-TI)
120 K=SQR(3)*256
130 K1=3.14/3:K2=3.14/6:K3=6/3.14:J=511
140 FORL=0TO3
150 C(L)=INT(RND(1)*8*(ABS(L-1.5)+.5))
160 NEXT
170 FORQ=1TO10
180 FORL=0TO1
190 A(L)=RND(1)*K1
200 D(L)=J*RND(1)-(1-ABS(A(L)-K2)*K3)*69
210 NEXT
220 C=INT(RND(1)*4)
230 IFC>1THENC(C)=INT(RND(1)*8*(C-1))
240 COLORC(0),C(1),C(2),C(3)
250 FORL=0TO5
260 A=A(0)
270 IFLAND1THENA=K1-A
280 X(L)=J+D(0)*COS(A+L*K1)
290 Y(L)=J+D(0)*SIN(A+L*K1)
300 A=A(1):IFLAND1THENA=K1-A
310 Y1(L)=J+D(1)*SIN(A+L*K1)
320 X1(L)=J+D(1)*COS(A+L*K1)
330 NEXT
340 FORL=0TO5
350 DRAWC,X(L),Y(L)TOX1(L),Y1(L)
360 NEXT
370 GETA$:IFA$<>""THEN:GRAPHIC4:END
390 NEXT:GOTO140
```

```
10 REM --- IPNOSI -----
15 XC=500:YC=500
20 DIM C(3)
40 FOR L=0TO3
50 GRAPHIC1
60 FORL=0TO3
70 C(L)=INT(RND(1)*8*((L/2()INT(L/2))+2))
80 NEXT
90 COLORC(0),C(1),C(2),C(3)
100  XD=INT(RND(1)*511)
110  YD=INT(RND(1)*511)
120  C=INT(RND(1)*4)
130  FORL=0TO3.14STEP3.14/60
140  X1=XC+XD*COS(L)
150  Y1=YC+YD*SIN(L)
160  X2=XC-XD*COS(L)
170  Y2=YC-YD*SIN(L)
180  DRAWC,X1,Y1TOX2,Y2
190  GETA$
200  IFA$)""THEN:GRAPHIC4:END
210  NEXT:GOTO60
```

**Appendice 7**

# T A V O L E  D E L  S I S T E M A  O P E R A T I V O

| | |
|---|---|
| C000-C045 | INDIRIZZO DI AZIONE PER TASTIERA |
| C046-C073 | INDIRIZZI PER LE FUNZIONI |
| C074-C091 | INDIRIZZI PER GLI OPERARTORI |
| C092-C192 | PAROLE CHIAVI DEL BASIC |
| C193-C2A9 | MESSAGGI DI ERRORE |
| C38A-C3B7 | RICERCA DELLO STACK PER FOR E GOSUB |
| C3B8-C3FA | APERTURA SPAZIO IN MEMORIA |
| C3FB-C407 | TEST PER LO STACK |
| C408-C434 | CONTROLLO DELLA MEMORIA DISPONIBILE |
| C435 | MANDA UN MESSAGGIO DI ERRORE |
| C474-C482 | STAMPA READY |
| C483-C532 | MANOVRA DI UN NUOVA LINEA DA TASTIERA |
| C533-C55F | RICOSTRUZIONE CONCATENATA DI UNA LINEA BASIC |
| C560-C57B | RICEVE LINEA DALLA TASTIERA |
| C57C-C612 | COMPATTAZIONE COMANDI BASIC |
| C613-C641 | RICERCA UN NUMERO DI LINEA BASIC DATO |
| C642 | ESEGUEIL COMANDO NEW |
| C660-C68D | ESEGUE CLR |
| C68E-C69B | RESET DEL BASIC ALLA PARTENZA |
| C69C-C741 | ESEGUE LIST |
| C742-C7EC | ESEGUE FOR |
| C7ED-C81C | ESEGUE UN COMANDO BASIC |
| C81D-C82B | ESEGUE RESTORE |
| C82C-C856 | ESEGUE STOP E END |
| C857-C870 | ESEGUE CONT |
| C871-C882 | ESEGUE RUN |
| C883-C89F | ESEGUE GOSUB |
| C8A0-C8D1 | ESEGUE GOTO |
| C8D2-C8EA | ESEGUE RETURN |
| C8EB-C905 | ESEGUE DATA |
| C906-C908 | RICERCA DEL PROSSIMO COMANDO BASIC |
| C909-C927 | RICERCA LA PROSSIMA LINEA BASIC |
| C928-C93A | ESEGUE IF |
| C93B-C94A | ESEGUE REM |
| C94B-C96A | ESEGUE ON |
| C96B-C9A4 | RIPORTA UN NUMERO FIXED POINT DAL BASIC |
| C9A5-CA1C | ESEGUE LET (PARTE PRIMA DELLA ROUTINE) |

| | |
|---|---|
| CA1D-CA2B | AGGIUNGE UN DIGIT ASCII ALL' ACCUMULATORE N 1 |
| CA2C-CA7F | ESEGUE LET (PARTE SECONDA) |
| CA80-CA85 | ESEGUE PRINT£ |
| CA86-CA99 | ESEGUE CMD |
| CA9A-CB1D | ESEGUE PRINT |
| CB1E-CB3A | STAMPA UNA STRINGA DALLA MEMORIA |
| CB3B-CB4C | STAMPA UN CARATTERE IN SINGOLO FORMATO (spazio,' |
| CB4D-CB7A | CONTROLLA UN INPUT ERRATO |
| CB7B-CBA4 | ESEGUE GET |
| CBA5-CBBE | ESEGUE INPUT No |
| CBBF-CBF8 | ESEGUE INPUT |
| CBF9-CC05 | PREPARA PER RICEVERE INPUT |
| CC06-CCFB | ESEGUE READ |
| CCFC-CD1D | EXTRA IGNORED E REDO FROM START |
| CD1E-CD77 | ESEGUE NEXT |
| CD78-CD9D | CONTROLLA I DATA,STAMPA TYPE MISMATCH |
| CD9E-CEF0 | INPUT E VALUTAZIONE DI ESPRESS. NUMERICHE O NON |
| CEF1-CEF6 | VALUTA LE ESPRESSIONI DENTRO LE () |
| CEF7-CEF9 | CONTROLLO PARENTESI DESTRE |
| CEFA-CEFC | CONTROLLA PARENTESI SINISTRE |
| CEFD-CF07 | CONTROLLO DELLA VIRGOLA |
| CF08-CF0C | STAMPA SYNTAX ERROR E ESCE |
| CF0D-CF13 | LIBERO PER FUTURE VALUTA DI FUNZIONI |
| CF14-CFA6 | CERCA IL NOME DELLA VARIABILE |
| CFA7-CFE5 | IDENTIFICA I RIFERIMENTI DELLE FUNZIONI |
| CFE6-CFE8 | ESEGUE OR |
| CFE9-D015 | ESEGUE AND |
| D016-D07D | ESEGUE CONFRONTI, NUMERICI O STRINGA |
| D07E-D08A | ESEGUE DIM |
| D08B-D112 | RICERCA LA LOCAZIONE DI UNA VARIABILE IN MEMORIA |
| D113-D11C | CONTROLLA SE UN CARATTERE ASCII E' ALFABETICO |
| D11D-D193 | CREA UNA NUOVA VARIABILE BASIC |
| D194-D1A4 | SUBROT. PER PUNTATORI DI MATRICI |
| D1A5-D1A9 | 32768 IN BINARIO |
| D1AA-D1D0 | VALUATZIONE DI ESPRESSIONE PER INTERO POSITIVO |
| D1D1-D34B | TROVA O CREA UNA MATRICE |
| D34C-D37C | CALCOLO DELLA MATRICE |
| D37D-D390 | ESEGUE FRE |
| D391-D39D | CONVERTE FIXED POINT IN FLOATING POINT |
| D39E-D3A5 | ESEGUE POS |
| D3A6-D3B2 | CONTROLLA SE E' UN COMANDO DIRETTO POSSIBILE |

| | |
|---|---|
| D3B3-D3E0 | ESEGUE DEF |
| D3E1-D3F3 | CONTROLLA LA SINTASSI DI FN |
| D3F4-D464 | VALUATA FN |
| D465-D474 | ESEGUE STRö |
| D475-D486 | CALCOLA IL VETTORE STRINGA |
| D487-D4F3 | CONTROLLA LA STRINGA |
| D4F4-D525 | SUB. DI COSTRUZIONE VETTORE STRINGA |
| D526-D5BC | ROUTINE DELLA GARBAGE COLLECTION |
| D5BD-D605 | CONTROLLO DELLA STRING COLLETION |
| D606-D63C | COLLECT STRING |
| D63D-D679 | ESEGUE LA CONCATENAZIONE DI STRINGHE |
| D67A-D6A2 | METTE UNA STRINGA IN MEMORIA |
| D6A3-D6DA | SCARTA UNA STRINGA NON CERCATA |
| D6DB-D6EB | PULOSCE LO STACK DESCRIPTOR |
| D6EC-D6FF | ESEGUE CHRö |
| D700-D72B | ESEGUE LEFTö |
| D72C-D736 | ESEGUE MIDö |
| D761-D77B | CARICA I PARAMETRI DI UNA STRINGA |
| D77C-D781 | ESEGUE LEN |
| D782-D78A | PASSA DA MODO STRINGA A MODO NUMERICO |
| D78B-D79A | ESEGUE ASG |
| D79B-D7AC | INPUT PER I PARAMETRI DEL BYTE |
| D7AD-D7EA | ESEGUE VAL |
| D7EB-D7F6 | DA DUE PARAMETRI PER POKE O WAIT |
| D7F7-D80C | CONVERTE IL FLOATING POINT IN FIXED POINT |
| D80D-D823 | ESEGUE PEEK |

| | |
|---|---|
| D824-D82C | ESEGUE POKE |
| D82D-D848 | ESEGUE WAIT |
| D849-D84F | AGGIUNGE .5 ALL' ACCUMULATORE N 1 |
| D850-D861 | ESEGUE LA SOTTRAZIONE |
| D862-D946 | ESEGUE ADDIZIONE |
| D947-D97D | COMPLEMENTO ALL' ACCUMULATORE N 1 |
| D97E-DP82 | STAMPA OVERFLOW E ESCE |
| D983-D9BB | SUBROU DI MULTIPLY A BYTE |
| D9BC-D9E9 | COSTANTI NELLE FUNZIONI |
| D9EA-DA2F | ESEGUE LOG |
| DA30-DA58 | ESEGUE LA MOLTIPLICAZIONE |
| DA59-DA8B | SUBROUTINE DI MOLTIPLICAZIONE BIT |
| DA8C-DAB6 | CARICA L' ACCUMULATORE N 2 |
| DAB7-DAD3 | TEST E AGGIUSTAGGIO DEGLI ACCUMULATORI 1 E 2 |
| DAD4-DAE1 | CONTROLLO DI OVERFLOW E UNDERFLOW |
| DAE2-DAF8 | MOLTIPLICA PER 1 |
| DAF9-DAFD | 10 IN FLOATYNG BINARIO |
| DAFE-DB06 | DIVIDE PER 10 |
| DB07-DB11 | ESEGUE DIVIDE ENTRO |
| DB12-DBA1 | ESEGUE DIVIDE PER |
| DBA2-DBC6 | CARICA L' ACC. N 1 DALLA MEMORIA |
| DBC7-DBFB | IMMAGAZZINA IN MEMORIA L'ACC. N 1 |
| DBFC-DC0B | COPIA L'ACCUMULATORE N 2 NELL 'ACC. 1 |
| DC0C-DC1A | COPIA L'ACCUMULATORE N 1 NELL' ACC. 2 |
| DC1B-DC2A | ARROTONDA L'ACCUMULATORE N 1 |
| DC2B-DC38 | CALCOLA IL VALORE SGN DELL' ACC. N 1 |
| DC39-DC57 | ESEGUE SGN |
| DC58-DC5A | ESEGUE ABS |
| DC5B-DC9A | CONFRONTA L' ACCUMULATORE 1 CON LA MEMORIA |
| DC9B-DCCB | CONVERTE UN NUMERO IN FLOATING IN FIXED POINT |
| DCCC-DCF2 | ESEGUE INT |
| DCF3-DD7D | CONVERTE UNA STRINGA IN FLOATING POINT |
| DD7E-DDB2 | DA UN NUOVO DIGIT ASCII |
| DDB3-DDC1 | COSTANTI DI CONVERSIONE STRINGA |
| DDC2 | STAMPA IN |
| DDCD-DDDC | STAMPA IL NUMERO DI LINEA BASIC |
| DDDD-DF10 | CONVERTE UN NUMERO O TIö IN ASCII |
| DF11-DF70 | COSTANTI PER CONVERSIONE NUMERICA |
| DF71-DF77 | ESEGUE SQR |
| DF78-DFB3 | ESEGUE UNA FUNZIONE POTENZA |
| DFB4-DFBE | ESEGUE NEGAZIONE BOLEANA |

| | |
|---|---|
| DFBF-DFEC | COSTANTI PER VALUTAZIONE STRINGA |
| DFED-E03F | ESEGUE EXP |
| E040-E089 | SUBROT NES DI VALUATZIONE DI FUNZIONI |
| E08A-E093 | MANIPOLAZIONI DI COSTANTI PER RND |
| E094-E0F5 | ESEGUE RND |
| E0F6-E260 | ROUTINES DI KERNAL |
| E261-E267 | ESEGUE COS |
| E268-E2B0 | ESEGUE SENO |
| E2B1-E2DC | ESEGUE TANGENTE |
| E2DD-E30A | COSTANTI PET VALUTAZIONE TRIG |
| E30B-E33A | ESEGUE ATN |
| E33B-E377 | COSTANTI PER VALUTAZIONI SERIE DI ATN |
| E378-E386 | INIZIALIZZA I VETTORI RAM |
| E387-E3A3 | SUBROUTINE PER RINVIO IN PAGINA ZERO |
| E3A4-E428 | INIZIALIZZAZIONE DEL SISTEMA |
| E429-E44E | BYTES FREE BBBB CBM BASIC V2BBBB |
| E44F-E47B | VETTORI DI INIZIALIZZAZIONE |
| E47C-E4FF | SPAZIO NON USATO |

# TAVOLA DELLE ROUTINE KERNAL

| | |
|---|---|
| E500-E504 | RIPORTO INDIRIZZO DEL 6522 |
| E505-E509 | RIPORTA IL MASSIMO DELLE RIGHE E COLONNE |
| E50A-E517 | LEGGE/STAMPA LA POSIZIONE DEL CURSORE |
| E518-E580 | INIZIALIZZA I/O |
| E581-E586 | HOME FUNCTION |
| E587-E5B4 | MUOVE IL CURSORE SECONDO IL PUNTATORE |
| E5B5-E5C2 | ENTRATA DEL NMI (TASTO RESTORE) |
| E5C3-E5CE | INIZIALIZZA IL 6561 |
| E5CF-E64E | TOGLIE IL CARATTERE DALLA CODA |
| E64F-E741 | INPUT DI LINEA DOPO RETURN |
| E742-E8E7 | STAMPA DI ROUTINE |
| E8E8-E8F9 | CONTROLLO DECREMENTO NEL PUNTATORE DI LINEA |
| E8FA-E911 | CONTROLLO INCREMENTO NEL PUNTATORE DI LINEA |
| E912-E928 | CONTROLLO COLORE |
| E929-E974 | TAVOLA DI CONVERSIONE DA CODICE SCHERMO A ASCII |
| E975-EAA0 | ROUTINES DI SCROLL DELLO SCHERMO |
| EAA1-EB1D | ROUTINE DI IRQ |
| EB1E-EC45 | ANALISI GENERALE DI TASTIERA |
| EC46-EE13 | TAVOLE DELLE MATRICI PER LA TASTIERA |
| EE14-EEBF | COMANDO AL BUS SERIALE DELLA PERIFERICA IN ATTESA |
| EEC0-EEC4 | INVIA L'INDIRIZZO SECONDARIO DOPO L'ASCOLTO |
| EEC5-EECD | RILASCIA LINEA ATTENTION DOPO L'ASCOLTO |
| EECE-EEE3 | PARLA L' INDIRIZZO SECONDARIO |
| EEE4-EEF5 | USCITA POTENZIATA AL BUS SERIALE |
| EEF6-EF03 | MANDA UN COMANDO UNTALK AL BUS SERIALE |
| EF04-EF18 | MANDA UN OMANDO UNLISTEN AL BUS SERIALE |
| EF19-EFA2 | INPUT DI UN BYTE DAL BUS SERIALE |
| EFA3-EFED | ROUTÎNE CONTINUA DI NMI |
| EFEE-F035 | TRASMETTE UN BYTE |
| F036-F173 | ROUTINE NMI PER RAGGRUPPARE I DATA NEI BYTES(RS232) |
| F174-F1E1 | MESSAGGI KERNAL |
| F1E2-F1F4 | SCRIVE UN MESSAGGIO SULLO SCHERMO |
| F1F5-F20D | PRENDE UN CARATTERE DA UN CANALE |
| F20E-F279 | INPUT DI UN CARATTERE DAL CANALE |
| F27A-F2C6 | USCITA DI UN CARATTERE A UN CANALE |
| F2C7-F308 | APRE UN CANALE PER INPUT |
| F309-F349 | APRE UN CANALE PER OUTPUT |

```
1  REM----------------
2  REM----------------
3  REM- DISASSEMBLER -
4  REM----------------
5  REM----------------
7  PRINT"CLR/HOME":GOSUB119
8  DATABRK,ORA,?,?,?,ORA,ASL,?
9  DATAPHP,ORA,ASL,?,?,ORA,ASL,?
10 DATABPL,ORA,?,?,?,ORA,ASL,?
11 DATACLC,ORA,?,?,?,ORA,ASL,?
12 DATAJSR,AND,?,?,BIT,AND,ROL,?
13 DATAPLP,AND,ROL,?,BIT,AND,ROL,?
14 DATABMI,AND,?,?,?,AND,ROL,?
15 DATASEC,AND,?,?,?,AND,ROL,?
16 DATARTI,EOR,?,?,?,EOR,LSR,?
17 DATAPHA,EOR,LSR,?,JMP,EOR,LSR,?
18 DATABVC,EOR,?,?,?,EOR,LSR,?
19 DATACLI,EOR,?,?,?,EOR,LSR,?
20 DATARTS,ADC,?,?,?,ADC,ROR,?
21 DATAPLA,ADC,ROR,?,JMP,ADC,ROR,?
22 DATABVS,ADC,?,?,?,ADC,ROR,?
23 DATASEI,ADC,?,?,?,ADC,ROR,?
24 DATA?,STA,?,?,STY,STA,STX,?
25 DATADEY,?,TXA,?,STY,STA,STX,?
26 DATABCC,STA,?,?,STY,STA,STX,?
27 DATATYA,STA,TXS,?,?,STA,?,?
```

```
28 DATALDY,LDA,LDX,?,LDY,LDA,LDX,?
29 DATATAY,LDA,TAX,?,LDY,LDA,LDX,?
30 DATABCS,LDA,?,?,LDY,LDA,LDX,?
31 DATACLV,LDA,TSX,?,LDY,LDA,LDX,?
32 DATACPY,CMP,?,?,CPY,CMP,DEC,?
33 DATAINY,CMP,DEX,?,CPY,CMP,DEC,?
34 DATABNE,CMP,?,?,?,CMP,DEC,?
35 DATACLD,CMP,?,?,?,CMP,DEC,?
36 DATACPX,SBC,?,?,CPX,SBC,INC,?
37 DATAINX,SBC,NOP,?,CPX,SBC,INC,?
38 DATABEQ,SBC,?,?,?,SBC,INC,?
39 DATASED,SBC,?,?,?,SBC,INC,?
40 DIMT$(255):FORI=0TO255:READT$(I):NEXT
41 INPUT"DEC. ADDRESS";A
42 PRINT"CLR/HOME
43 FORM=1TO11
44 I=PEEK(A)
45 K=(I)AND15
46 IFI=0THENL=1:GOTO65
47 IFK=8THENL=1:GOTO65
48 IFK=10THENL=1:GOTO65
49 IFK<>0THEN53
50 IFI=32THEN57
51 IF(I)AND16THEN53
52 IFI<128THENL=1:GOTO65
53 IFK=0THEN61
54 IFK=9THEN58
```

```
55 IFK<10THEN64
56 J=PEEK(I)
57 L=3:GOTO65
58 K=(I)AND16
59 IFK=0THEN64
60 GOTO57
61 K=(I)AND240
62 IFK>111THEN64
63 IFK=32THEN57
64 L=2
65 N=A:GOSUB110
66 PRINT"    ";
67 IFT$(I)="?"THENL=1
68 FORX=ATOA+L-1
69 N=PEEK(X):GOSUB110:PRINT" ";
70 NEXT
71 PRINT:PRINT"(CRSR/UP)(10CRSR/RIGHT)"
72 IFT$(I)<>"?"THEN74
73 PRINTCHR$(13)"(REVERSE).BYT   $";:N=I:L=1:GOSUB110:GOTO101
74 PRINTCHR$(13)"(REVERSE)"T$(I);
75 IFL=1THEN101
76 PRINT"   ";
77 X=(I)AND15:Y=(I)AND240
78 IFL=3THEN92
79 ONX+1GOTO80,86,85,10000,89,89,89,10000,10000,85
80 IF(Y)AND16THEN82
81 GOTO85
```

```
82 N=PEEK(A+1):IFN)127THENN=-(256-N)
83 PRINT"$";:N=A+N+2
84 GOSUB110:GOTO101
85 PRINT"#$";:N=PEEK(A+1):GOSUB110:GOTO101
86 PRINT"($";:N=PEEK(A+1):GOSUB110
87 IF(Y)AND16THENPRINT"),Y";:GOTO101
88 PRINT",X)";:GOTO101
89 PRINT"$";:N=PEEK(A+1):GOSUB110
90 IF(Y)AND16THENPRINT",X";
91 GOTO101
92 ONX+1GOTO95,8,8,8,8,8,8,8,8,93,94,94
93 PRINT"$";:N=PEEK(A+2):GOSUB110:N=PEEK(A+1):GOSUB110:PRINT",Y";:GOTO
94 IFI=200THEN93
95 PRINT"$";:N=PEEK(A+2):GOSUB110:N=PEEK(A+1):GOSUB110
96 IF(Y)AND16THENPRINT",X";
97 GOTO101
98 IFI=198THEN95
99 IFI()108THEN95
100 PRINT"($";:N=PEEK(A+2):GOSUB110:N=PEEK(A+1):GOSUB110:PRINT")";
101 PRINT:PRINT"(CRSR/UP)(10CRSR/RIGHT)"
102 FORJ=1TOL
103 X=PEEK(A+J-1)
104 IFX(32THENX=X+32
105 IFX)127THENX=X-128:GOTO104
106 PRINTCHR$(X);:NEXT
```

```
107 A=A+L
108 PRINT:  NEXT
109 GOTO41
110 A$="Ø123456789ABCDEF"
111 H$=""
112 K=N-INT(N/16)*16
113 N=INT(N/16)
114 H$=MID$(A$,K+1,1)+H$
115 IFN>ØTHEN112
116 IFLEN(H$)=1THENPRINT"Ø";
117 PRINTH$;
118 RETURN
119 FORI=1TO2ØØØ
120 PRINT"(REVERSE)DISASSEMBLATORE(3CRSR/DOWN)"
121 RETURN
```

ROUTINE DI TRASFERIMENTO IN
MEMORIA DI 256 CARATTERI

DA  LOC. DI PARTE.
TO  LOC. DI ARRIVO

```
        LDX #$ 02
   LOOP LDA $ DA  X
        STA $ TO  X
        INX
        BNE $ LOOP
        RTS
```

ROUTINE DI TRASFERIMENTO SU VIDEO
DI 256 CARATTERI
USANDO L'INDIRIZZAMENTO
INDICIZZATO INDIRETTO

```
3000      LDA #$ LOW
3002      STA $ 24
3004      LDA #$ HIGH
3006      STA $ 25
3008      LDY #$ 00
300A LOOP LDA ($24),Y
300C      STA $ VIDEO,Y
300F      LDA #$ 02
3011      STA $ COLORE,Y
3014      BNE $ LOOP
401E      RTS
```

? SUBROUTINE DI STAMPA DI UNA STRINGA
IN MEMORIA

? ORIZZ POSIZIONE DEL CURSORE IN ORIZZONTALE
VERT POSIZIONE DEL CURSORE IN VERTICALE
INIZ LOCAZIONE DI INIZIO DELLA STRINGA
NUM NUMERO DEI CARATTERI DELLA STRINGA

LOW PARTE BASSA DELL'INDIRIZZO
HIGH PARTE ALTA DELL'INDIRIZZO
VIDEO INDIRIZZO DELLE LOC. VIDEO
COLORE INDIRIZZO DELLE LOC. DEL COLORE

```
        LDA #$ 93
        JSR $ E742
        NOP
        LDY #$ ORIZZ
        LDX #$ VERT
        CLC
        JSR $ FFF0
        NOP
        LDY #$ 00
STAMPA  LDA $ INIZ,Y
        JSR $ E742
        INY
        CPY #$ NUM.
        BNE $ STAMPA
        RTS
```

NOME ____________________________________________

VIA ____________________________________________

CITTA' ____________________________________________

SONO IN POSSESSO DI ____________________________________________

CON LE SEGUENTI PERIFERICHE ____________________________________________

____________________________________________

[ ] Desidero ricevere gratuitamente il Vostro catalogo ed essere inserito nell' E.V.M. MAILING LIST

[ ] Desidero informazioni sulla SOFTWARE BANK

SEGNALAZIONE DI ERRORI ____________________________________________

____________________________________________

____________________________________________

SPEDIRE IN BUSTA A:

E.V.M Computers
Via Marconi 9/a
52025 MONTEVARCHI (AR)